Constrained RESTful Environments WG (core)

Chairs:

Cullen Jennings <fluffy@cisco.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**
- **Be aware of the IPR principles, according to RFC 3979 and its updates**

✓Blue sheets
✓Scribe(s)

Milestones (from WG charter page)

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

Document submissions to IESG:

- **Apr 2010 Select WG doc for basis of CoAP protocol**
- **Dec 2010 1 – CoAP spec$^+$ with mapping to HTTP REST submitted to IESG as PS**
- **Dec 2010 2 – Constrained security bootstrapping spec submitted to IESG as PS**

- **Jan 2011 Recharter to add things reduced out of initial scope**

link-format-11 in IESG

- **2012-03-15 IESG Evaluation::AD Followup**
- **2012-02-28 IESG Evaluation**
- **2012-02-14 IETF Last Call**

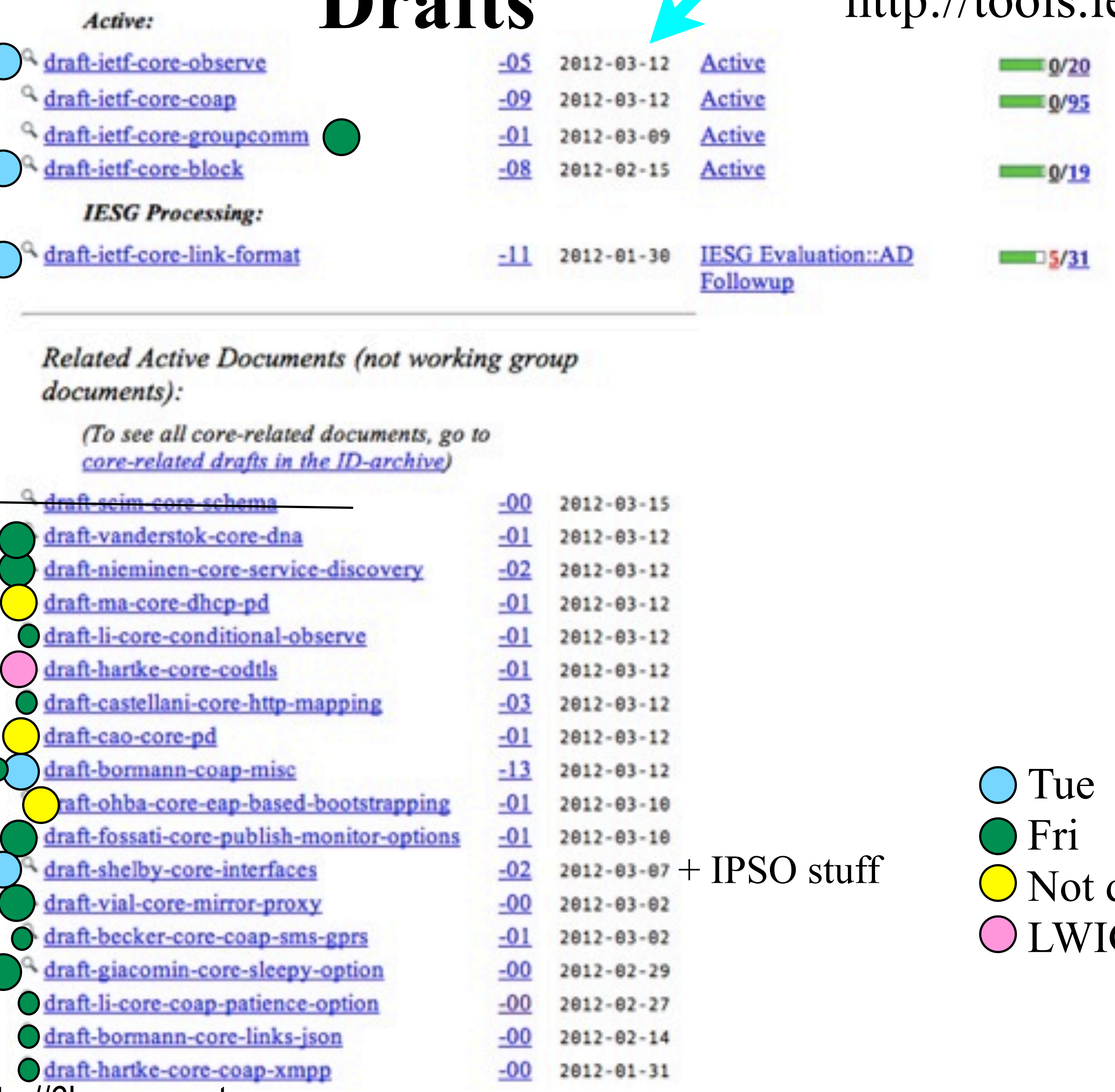
- **2 DISCUSSes (Jari Arkko: ABNF, Russ Housley: reg.)**
- ***Discuss* today**

coap-09, block-08, observe-05 in WGLC

- WGLC issued on March 20
  - now, everybody read it again!
  - send email to list,
    including draft name and a name for the issue
  - any outstanding IPR declarations?
  - how is your implementation going?
- time for good review, IETF, holidays
  ➔ long WGLC until April 16

Drafts

http://tools.ietf.org/wg/core/



Active:

🔵 🔵	draft-ietf-core-observe	-05	2012-03-12	Active	0/20
🔵	draft-ietf-core-coap	-09	2012-03-12	Active	0/95
🟢	draft-ietf-core-groupcomm	-01	2012-03-09	Active	
🔵	draft-ietf-core-block	-08	2012-02-15	Active	0/19

IESG Processing:

🔵	draft-ietf-core-link-format	-11	2012-01-30	IESG Evaluation::AD Followup	5/31

Related Active Documents (not working group documents):

(To see all core-related documents, go to core-related drafts in the ID-archive)

	~~draft-scim-core-schema~~	-00	2012-03-15	
🟢	draft-vanderstok-core-dna	-01	2012-03-12	
🟢	draft-nieminen-core-service-discovery	-02	2012-03-12	
🟡	draft-ma-core-dhcp-pd	-01	2012-03-12	
🟢	draft-li-core-conditional-observe	-01	2012-03-12	
🟣	draft-hartke-core-codtls	-01	2012-03-12	
🟢	draft-castellani-core-http-mapping	-03	2012-03-12	
🟡	draft-cao-core-pd	-01	2012-03-12	
🟢 🔵	draft-bormann-coap-misc	-13	2012-03-12	
🟡	draft-ohba-core-eap-based-bootstrapping	-01	2012-03-10	
🟢	draft-fossati-core-publish-monitor-options	-01	2012-03-10	
🔵	draft-shelby-core-interfaces	-02	2012-03-07	+ IPSO stuff
🟢	draft-vial-core-mirror-proxy	-00	2012-03-02	
🟢	draft-becker-core-coap-sms-gprs	-01	2012-03-02	
🟢	draft-giacomin-core-sleepy-option	-00	2012-02-29	
🟢	draft-li-core-coap-patience-option	-00	2012-02-27	
🟢	draft-bormann-core-links-json	-00	2012-02-14	
🟢	draft-hartke-core-coap-xmpp	-00	2012-01-31	

🔵	Tue
🟢	Fri
🟡	Not discussed
🟣	LWIG (Thu)

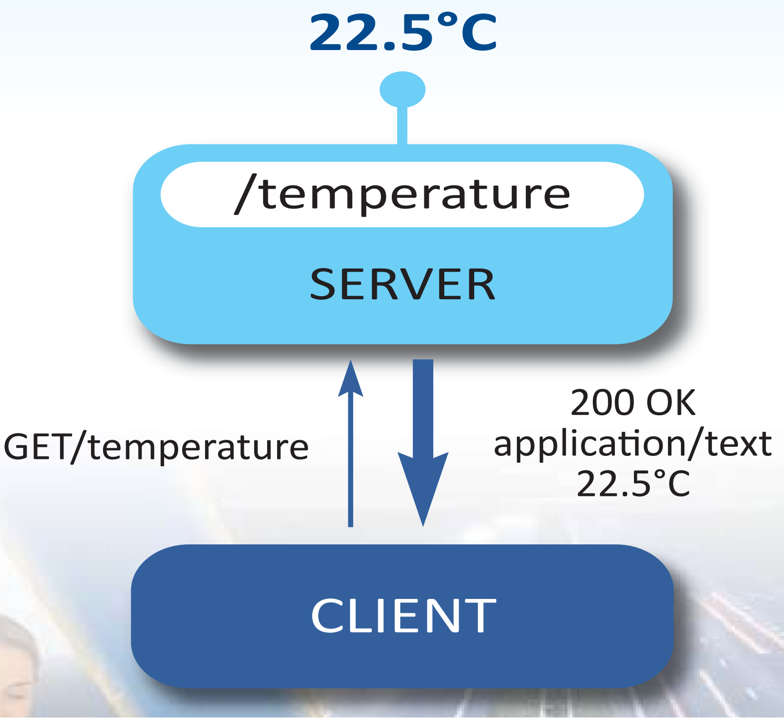
ETSI
World Class Standards

IoT CoAP Plugtests

24-25 March 2012, Paris

Registration Deadline: 9 March 2012
Website: http://www.etsi.org/plugtests

PLUGTESTS™
INTEROP EVENTS



ETSI Plugtests, the IPSO Alliance and the FP7 Probe-IT project are pleased to invite you to participate in the first Internet of Things CoAP Plugtest, taking place from 24-25th March 2011 in Paris, France.
The event is co-located with the 83rd IETF held March 26-30th.

ETSI

World Class Standards

PRELIMINARY RESULTS OF 1ST COAP PLUGTEST

Sebastian Müller, Technical Coordinator, ETSI

Slight Mods by Carsten Bormann

Agenda

- What is a Plugtest?
- Interoperability Test Procedure
- Reflection on CoAP Plugtest
- Participants
- Plugtest Results
- Conclusion

What is a Plugtests event?

- A test event organized and run by a neutral body
  - Scope, test infrastructure and test scenarios based on standard
  - Scheduling
  - Test Results and Feedback to Standards Development
- An opportunity for engineers
  - Evaluate the interoperability of their products
  - Validate their understanding of the base specification
  - Save time
- An opportunity for vendors
  - To demonstrate end 2 end interoperability to operators/end customers
  - Promote the technology and community
- An opportunity for Standards Development
  - Gaps, ambiguities, interpretations
  - A tool to validate and enhance the quality of standards

Interoperability Test Procedure

- Connect client and server over test network
- Check connectivity between devices
- Perform tests according to Plugtest Guide
  - Check if test runs to completion
  - Check results from an interoperability point of view: Is the intended result visible at the application layer?
- Result determination and reporting
  - Result OK: run next test
  - Result NOK: check monitor tools to identify source of error
  - Report results in ETSI Test Reporting Tool

Reflection on the CoAP Plugtest (1/2)

- Jointly organized by ETSI, ProbeIT, IPSO Alliance
- Hosted at IETF#83
- 2 day event
- Sponsored by EC
- Test specification produced by ETSI and ProbeIT
  - Distributed 2 months prior to event
  - Total of 26 tests
- ETSI Tools
  - WIKI
  - Scheduling Tool
  - Test Reporting Tool
- IRISA tool – Passive Trace Validation
- BUPT tool – Lossy Gateway

Reflection on the CoAP Plugtest (2/2)

- Active involvement by all players in build-up to CoAP Plugtest through 3 conference calls and email reflector
  - Thanks to all participants for reviewing the test specification and helping to correct errors/ambiguities
- Test sessions for IOP assessment followed by selective wrap-up for main interop points of the day
- Demo of 6LowPAN Conformance Tests
- Good Community spirit
- Good industry participation
  - 15 companies with implementations
  - 4 companies as part of plugtest team
  - More than 50 people
- Important mix of technologies
  - 6 different embedded wireless platforms; TinyOS, Contiki, Custom OS; C, C++, Java, C#, Ruby, JavaScript

Participants

#	*Implementations*
1	Actility
2	Watteco
3	ETH Zürich*)
4	Hitachi
5	Huawei
6	Intecs
7	KoanLogic
8	Patavina
9	Sensinode
10	Uni Bremen*)
11	Uni Rostock
12	Rtx
13	Ibbt
14	Ferrara
15	(Mystery)

#	*Plugtest Team*
1	IRISA
2	BUPT
3	CATR
4	ETSI

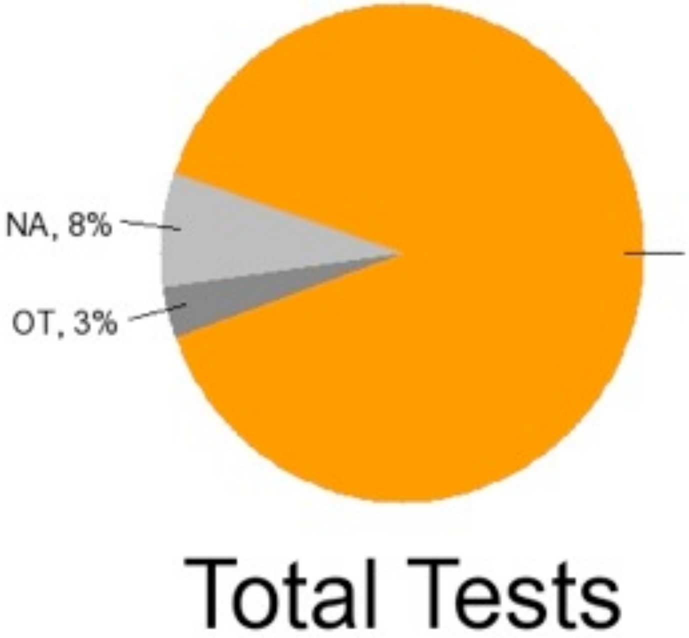
*) Present with multiple implementations (total of 7)

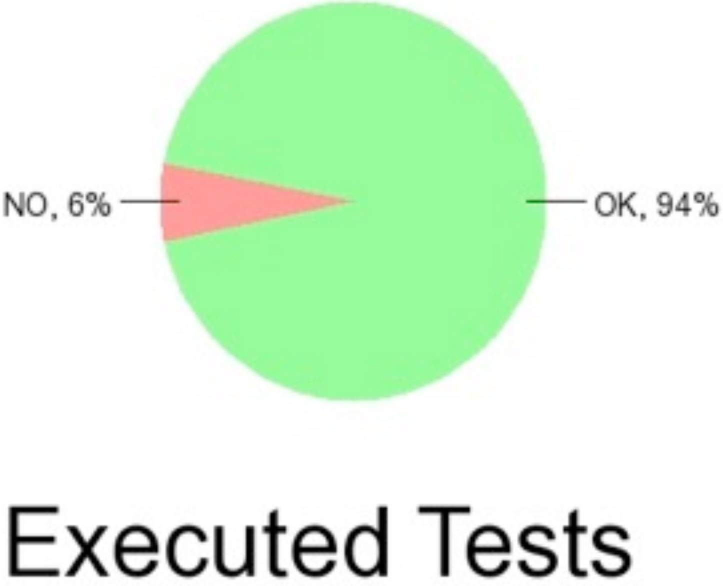
Scope of Interoperability Tests

- CORE
  - Get, Post , Put, Delete, Token, Uri Path/Query
  - Lossy context
- LINK
- BLOCK
- OBSERVE
  - Resource Observation
  - Deregistration Detection

- Test Spec
  - 27 tests
  - Structured in optional/ mandatory
  - 16 CORE
  - 2 LINK
  - 4 BLOCK
  - 5 OBSERVE

Test Results – Overview

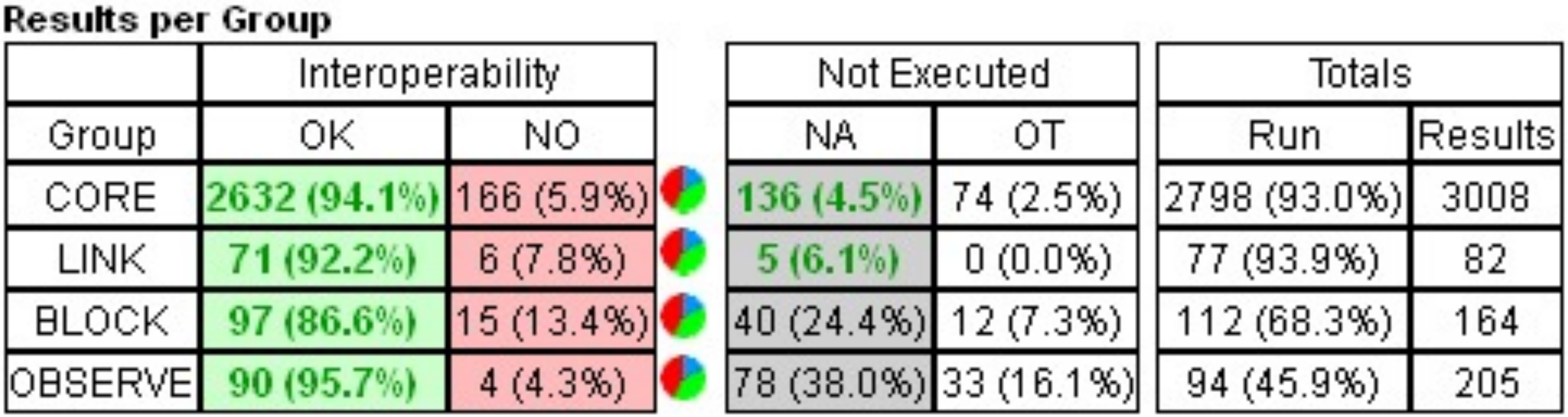




Analysis – Mandatory Tests

- More than 3000 tests executed
- More than 90 % of executed tests passed
  - High level of interoperability
- 8 % of the tests are not executed due to non implemented features
  - Mainly BLOCK and OBSERVE
- 3% of the tests not executed due to time limitation

Test Results – Per Group



Results per Group

	Interoperability			Not Executed		Totals	
Group	OK	NO		NA	OT	Run	Results
CORE	2632 (94.1%)	166 (5.9%)	●	136 (4.5%)	74 (2.5%)	2798 (93.0%)	3008
LINK	71 (92.2%)	6 (7.8%)	●	5 (6.1%)	0 (0.0%)	77 (93.9%)	82
BLOCK	97 (86.6%)	15 (13.4%)	●	40 (24.4%)	12 (7.3%)	112 (68.3%)	164
OBSERVE	90 (95.7%)	4 (4.3%)	●	78 (38.0%)	33 (16.1%)	94 (45.9%)	205

Blocking issues

- Token Options (often implemented only partially)
- Block1 option (i.e., blockwise PUT/POST)
- Clients, having received an incoming packet , must use in their response the IP address to which the incoming packet has been addressed; Clients shall not change their source address in a response

- Suggestion: Client should not always use default port (src port == 5683) as source port for requests. Ephemeral port range should be used to make sure that hard coded addresses are not used

Conclusion on Event

- Well prepared event
  - Participants were prepared as the test spec was delivered well in advance
  - Stable Test Spec (no errors reported during the plugtest)
  - Stable test infrastructure
  - Pre testing was very useful
- Everybody was able to execute against a fair number of other companies
- All tests defined could be executed in a single 1 hour session
  - An initial setup time of at least 1 hour would be beneficial
- Interest in conformance tests
- Plugtest enabled to resolve bugs and to achieve higher quality implementations
  - Some bugs were fixed in each implementation

Conclusion on the Results

- Implementations have been all compatible on the basic level
  - Sent data could be decoded and interpreted properly by receivers
  - Vast majority of equipment performed well
- Mature and prototype implementations exist
  - The difference between mature and prototype implementations is in the level of coverage of implemented features
  - When features are implemented, then high interoperability is observed
  - Conformance monitoring shows that more conformance testing is needed
- COAP base standards are mature
  - This applies to the parts of base standard that were covered in the plugtest
- This first plugtest is a success with regards to the number of participants and the test results
  - Vendors were mature enough to start with interoperability testing
  - This event is a clear signal to the community about the usefulness of testing

What is next?

- To organize another Plugtests event in Q4 2012
  - Scope and location to be defined
- To include in scope tests for
  - Proxy
  - Security DTLS
  - IPSO profile
  - Full set of options
  - Resource Directory
- To consider a slightly longer event
- More conformance sessions during the Plugtests event

Link and contact

- Plugtest web page, Mailing list
  - http://www.etsi.eu/plugtests/coap/coap.htm

- For any information contact plugtests@etsi.org

83rd IETF: core WG Agenda

09:00	Introduction, Agenda, Status	Chairs (10)
09:10	SOS Ws Report	HT (10)
09:20	ETSI Plugtest Report	CB (10)
09:30	1 – core Link Format	ZS (90)
09:50	1 – core, block, observe, WGLC	ZS, KH, CB (20)
10:45	Interfaces	ZS (45)
11:30 retire to Friday, 09:00	Intro	Chairs (05)
11:30 retire		

Tue/Fri scheduling

- **Who will *not* be present on Friday?**

Group 0: @IESG link-format-11

CoRE Link Format (draft-ietf-core-coap-11)

IANA, IETF-LC & IESG Summary

Zach Shelby

CoRE WG, IETF-83 Paris

IANA Reviews

- Reviews Completed
  - IANA review – OK
  - /.well-known review – No objections to “core”
- Reviews in Progress
  - Media type review for application/link-format
  - Link relation review for “hosts”

IETF Last Call Comments

- Security Review – Richard Barnes
  - [#189] Additional text on access control
    - resource discovery might be gated through authz
  - [#190] Conversion from HTTP Link Header
    - explain LWS conversion
  - [#191] Origin definition from RFC6454
    - make use of useful reference
  - [#192] Query pattern matching (% encoding issues)
    - editorial
  - [#193] Anchor restriction for "hosts" relation
    - both ends of “hosts” relation are on one origin

IETF Last Call Comments

- Gen-Art Review – Joel Halpern
  - [#195] Create a registry for rt= and if= values
    - handle similar to Link Relation Registry
- App Review – Julian Reschke
  - [#194] Rules for determining the Context of a link document
    - editorial (use #191)
  - [#196] Clarify URI fetching rule for attribute values in Section 3
  - [#197] Upgrade to RFC5234 ABNF (lose LWS issue)
  - [#198] Always allow both token and quoted-string in attributes
  - [#199] Put multiple values in a single attribute, separated by spaces (do not allow multiple attributes)
  - [#200] Change “uri” in query string to “href” (like HTML <link>)

IESG Discusses

- Russ Housley
  - Resolve the rt= and if= registries as per Gen-Art review [#195]
- Jari Arkko
  - ABNF improvement suggestions (~ [#197] + more nits)

- Need to reply to the “comments”, too

Group 1: WGLC

coap-09, block-08, observe-05

core-coap-09

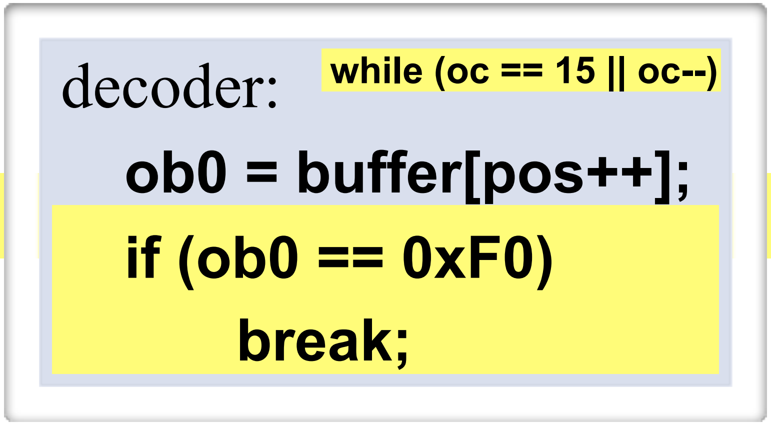
- changes in –09:
  - removed artificial limit for number of options (“oc=15”)
  - be more explicit about multicast implosion prevention
  - QoI comment about piggy-backing
  - track progress of TLS/DTLS raw public keys

coap-09: Seven lines of code for *oc=15*

```
b0 = 0x40 + (tt << 4);
buffer[0] = b0 + 15;

… encode options …

if (option_count >= 15 || first_fragment_already_shipped)
   buffer[pos++] = 0xF0;                    /* use delimiter */
else                                        /* save a byte: */
   buffer[0] = b0 + option_count;           /* backpatch */
```



```
decoder:    while (oc == 15 || oc--)
    ob0 = buffer[pos++];
    if (ob0 == 0xF0)
        break;
```

yellow = new/changed code

Closing the RPK provisioning story

- In Taipei we decided to focus on RawPublicKey mode
  - coap-08 integrated RawPublicKey support
  - The TLS WG produced draft-ietf-tls-oob-pubkey (thanks!)
  - coap-09 added an Appendix D on provisioning
- RawPublicKey mode is in solid shape
- What to do with the identifier (and thus Appendix D)?
  - Remove it and just use the RawPublicKey as an identifier

    + Short anyways for must-implement ECC Cipher, no hash needed

    + No need to reference external documents or define new hash functions
  - Define a new identifier hash function in Appendix D

    + Applicable to any kind of public key

    - This will be needed by other protocols, referencing CoAP is awkward

    - Would involve a delay in finishing this work

  1. Define a new identifier hash function in some other draft

     + Applicable to any kind of public key

     + Easily referenced by other protocols, useable with more than just DTLS

CoRE Observation

draft-ietf-core-observe-05

K. Hartke

Changes from -03 to -04

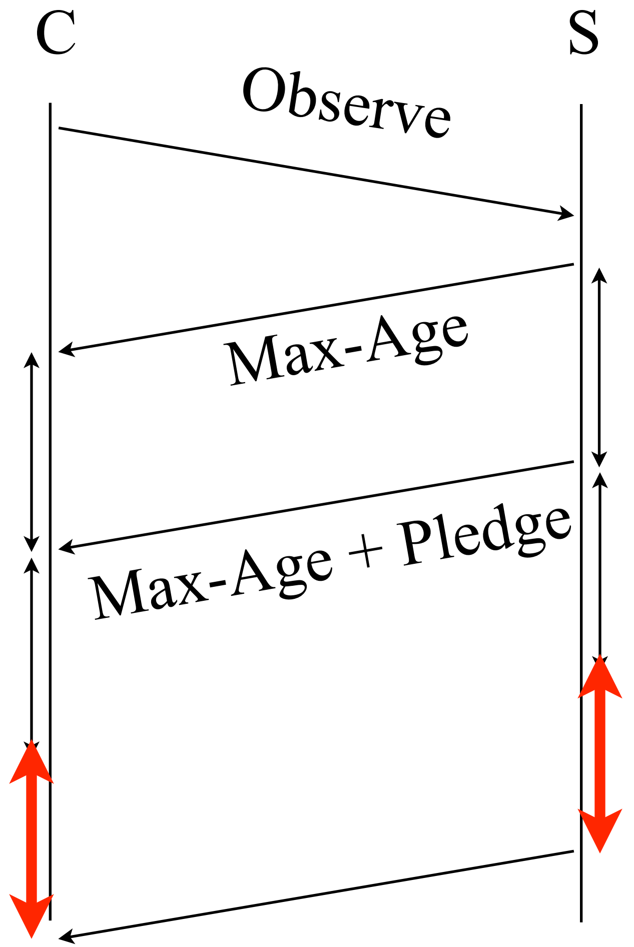
- Removed the "Max-OFE" Option again.
  - Solve the remaining problems with later extensions like Pledge
- Added a section on cancellation.
  - Allowed RST in reply to non-confirmable notifications.

Changes from -04 to -05

- Recommended not to re-register while a notification from the server is still likely to arrive (#174).
  (Avoid cross-over between the last notification and the client's request.)
- Relaxed requirements when reacting to RST in reply to NON notifications.
- Added an implementation note about careless GETs (#184).

Robust Observation Relationships (#174)

- We have the 80 % solution
- What about the other 20 %
  - Pledge?
  - Server-side Patience?
  - Need to work out exact impact in caching/intermediaries



License for waiting patiently

The block option

- **Some resource representations are > MTU bytes**
- **Transfer in blocks**

```
 0
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|blocknr|M| szx |
+-+-+-+-+-+-+-+-+
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        block nr       |M| szx |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 0                   1                   2
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                block nr               |M| szx |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

M: More Blocks
szx: $\log_2$ Blocksize – 4

Decisions:

- Block size is power of 2
- 16 ≤ Block size ≤ 2048

Status of core-block-08

- –08: Integrated the size option (18, uint, elective)
- Otherwise, no technical changes since the split Block1/Block2, editorial:
- –07: an example for blockwise POST
- –06: minor editorial
- –05: editorial rewrite concluded

Group 3: Interfaces

draft-shelby-core-interfaces-02

CoRE Interfaces

Zach Shelby, Matthieu Vial

CoRE WG, IETF-83 Paris

Prologue

- CoAP is just a web transfer protocol
- How do we use it for real applications?
  - What to put in link description fields?
  - How to design & interact with a resource?
  - How to represent data in payloads?
- Let’s make this easy, interoperable and efficient:
- CoRE Interfaces (draft-shelby-core-interfaces-02)
  - Basic REST interfaces & function set design for embedded devices & M2M applications
- The IPSO Profile
  - A set of generic function sets useful for IP smart objects
  - Developed for use in IPSO Alliance interop/demo events

CoRE Interfaces

- draft-shelby-core-interfaces-02
  - Applicable to both CoAP or HTTP
  - **Profiles**, made up of function sets
    - **Function sets**, made up of sub-resources
      - **Sub-resources** and their attributes
        - Path, resource type, interface type(s), data type etc.
  - Simple REST interfaces
    - Link list
    - Batch
    - Linked batch
    - Sensor
    - Parameter, Read-only Parameter
    - Actuator
    - Resource observation query interface

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 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

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
```

ipso Alliance

Enabling the Internet of Things

www.ipso-alliance.org

The IPSO Profile

- IPSO organizes and promotes smart object interoperability
- Recently a simple profile was developed:
    - For use in IPSO interop and demo events
    - To promote application level device interoperability
- The goals of this design was:
    - Meet the needs of embedded devices of IPSO members
    - Support use over both CoAP and HTTP
    - Compliment existing profiles like SE2.0 and oBIX
    - Aimed at general automation uses
- Will be used in multi-vendor demonstration April 3-4th in Paris
- The specification is available for download:

http://www.ipso-alliance.org/technical-information

Function Sets & Data Formats

- Design based on draft-shelby-core-interfaces
  - Uses the interface definitions and function set model
- Function Sets currently defined
  - Device (model, manufacturer etc.)
  - General Purpose IO
  - Power (power meters and relays)
  - Sensors (extensible)
  - Light Control (simple light control)
  - Message (status, alarms, displays)
  - Location (GPS and XY location)
- Data Formats
  - text/plain (xsd:string, xsd:boolean, xsd:integer, xsd:decimal)
  - application/senml+json

Example Device

Test server is on-line at coap://interop.ams.sensinode.com:8000

Web links of an example device:

```
</dev/mfg>;rt="ipso:dev-mfg",
</dev/ser>;rt="ipso:dev-ser",
</dev/mdl>;rt="ipso:dev-mod",
</pwr/0/w>;rt="ipso:pwr-w",
</pwr/0/kwh>;rt="ipso:pwr-kwh",
</pwr/0/rel>;rt="ipso:pwr-rel",
</pwr/1/w>;rt="ipso:pwr-w",
</pwr/1/kwh>;rt="ipso:pwr-kwh",
</pwr/1/rel>;rt="ipso:pwr-rel",
</gpio/btn/0>;rt="ipso:gpio-btn",
</lt/led0/on>;rt="ipso:lt-on",
</sen/temp>;rt="ucum:Cel";obs,
</sen/co2>;rt="ucum:ppm"
```

Example Requests

Device Function Set

```
Req: GET /dev/mfg (Accept: text/plain)
Res: 2.05 Content (text/plain)
   Body: IPSO Alliance
```

Power Function Set

```
Req: GET /pwr/0/w (Accept: text/plain)
Res: 2.05 Content (text/plain)
   Body: 123


Req: PUT /pwr/0/rel (Accept: text/plain)
   Body: 0
Res: 2.04 Changed (text/plain)


Req: GET /pwr/0/w (Accept: text/plain)
Res: 2.05 Content (text/plain)
   Body: 0
```

Group 2: groupcomm

Group Communication for CoAP

Akbar Rahman
Esko Dijk

IETF 83, March 2012

http://tools.ietf.org/html/draft-ietf-core-groupcomm-01

Introduction

- Document was adopted as a WG draft at IETF82 (Taipei)
- IP Multicast approach was the adopted mechanism for CoAP Group Communications
- Other alternatives, enhancements and background information is being maintained in dijk-core-groupcomm-misc
    - http://tools.ietf.org/html/draft-dijk-core-groupcomm-misc-00

CoAP Group Comm with IP Multicast

- CoAP sub-networks need to be connected directly to IP multicast enabled routers (e.g. running PIM-SM [RFC4601])

- Sending CoAP node (client) sends single message with IP address to selected multicast IP group address (and underlying IP Multicast routers will then distribute (multicast) the message)

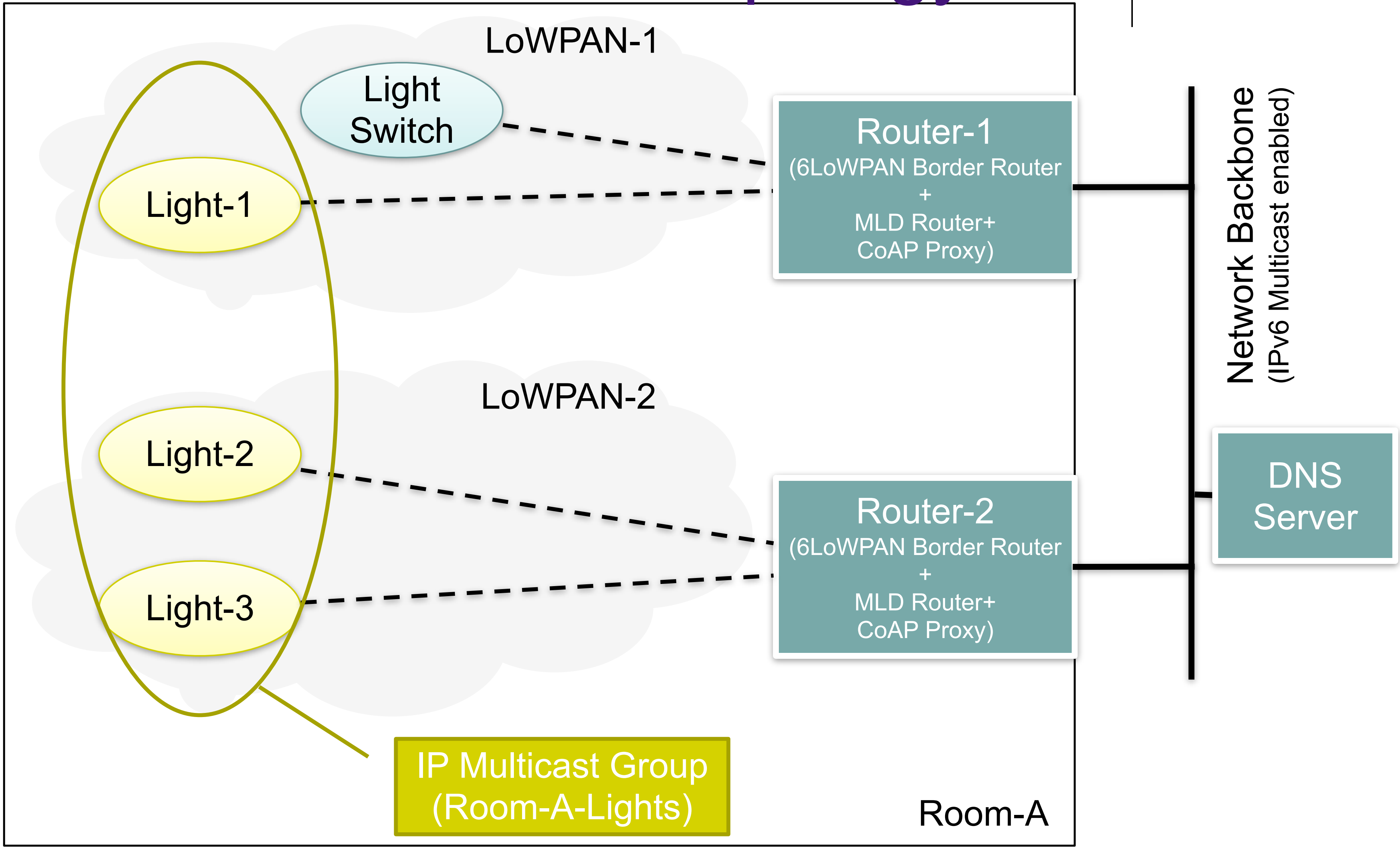
- Receiver CoAP nodes (servers) use MLD [RFC3810] to subscribe (and receive) any messages sent to selected IP multicast group

- Note: IP Multicast does NOT provide guaranteed delivery

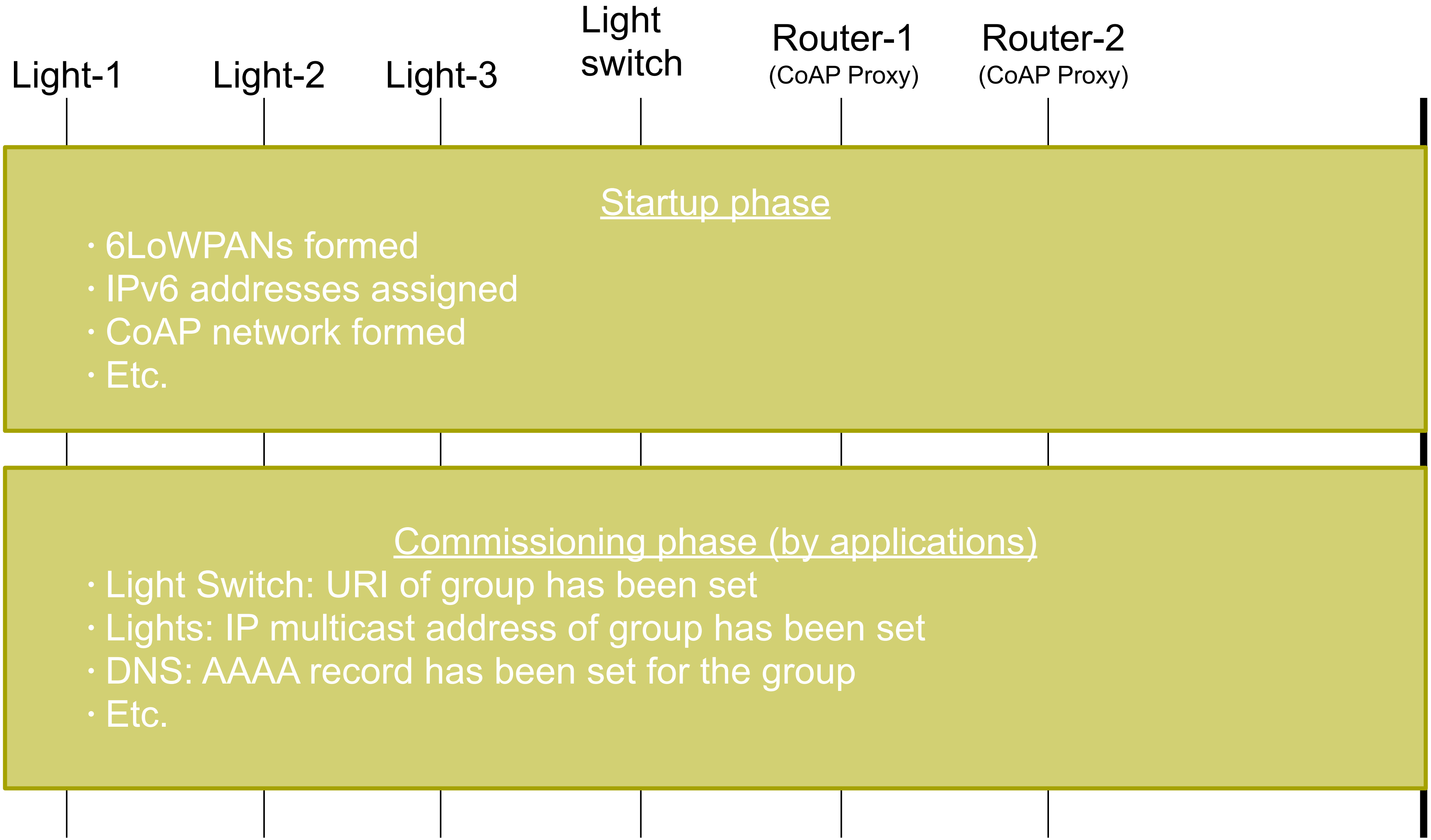
Use Case (and Example Protocol Flow)

TURNING ON LIGHTS IN A LARGE CONFERENCE ROOM

Room-A Network Topology



Turning on lights in Room-A (1/5)



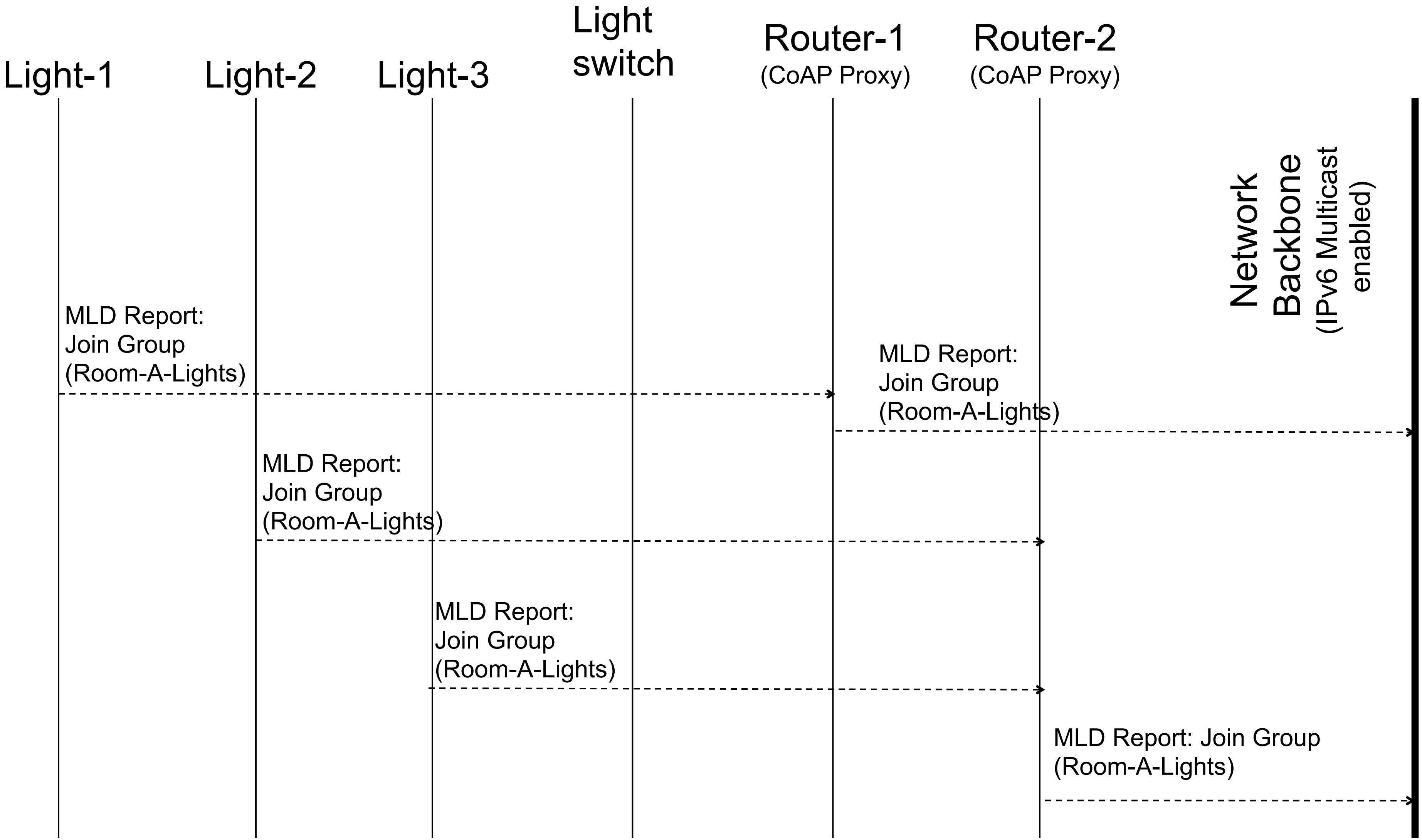
Startup phase

- 6LoWPANs formed
- IPv6 addresses assigned
- CoAP network formed
- Etc.

Commissioning phase (by applications)

- Light Switch: URI of group has been set
- Lights: IP multicast address of group has been set
- DNS: AAAA record has been set for the group
- Etc.

Turning on lights in Room-A (2/5)



Light-1

Light-2

Light-3

Light switch

Router-1 (CoAP Proxy)

Router-2 (CoAP Proxy)

Network Backbone (IPv6 Multicast enabled)

MLD Report: Join Group (Room-A-Lights)

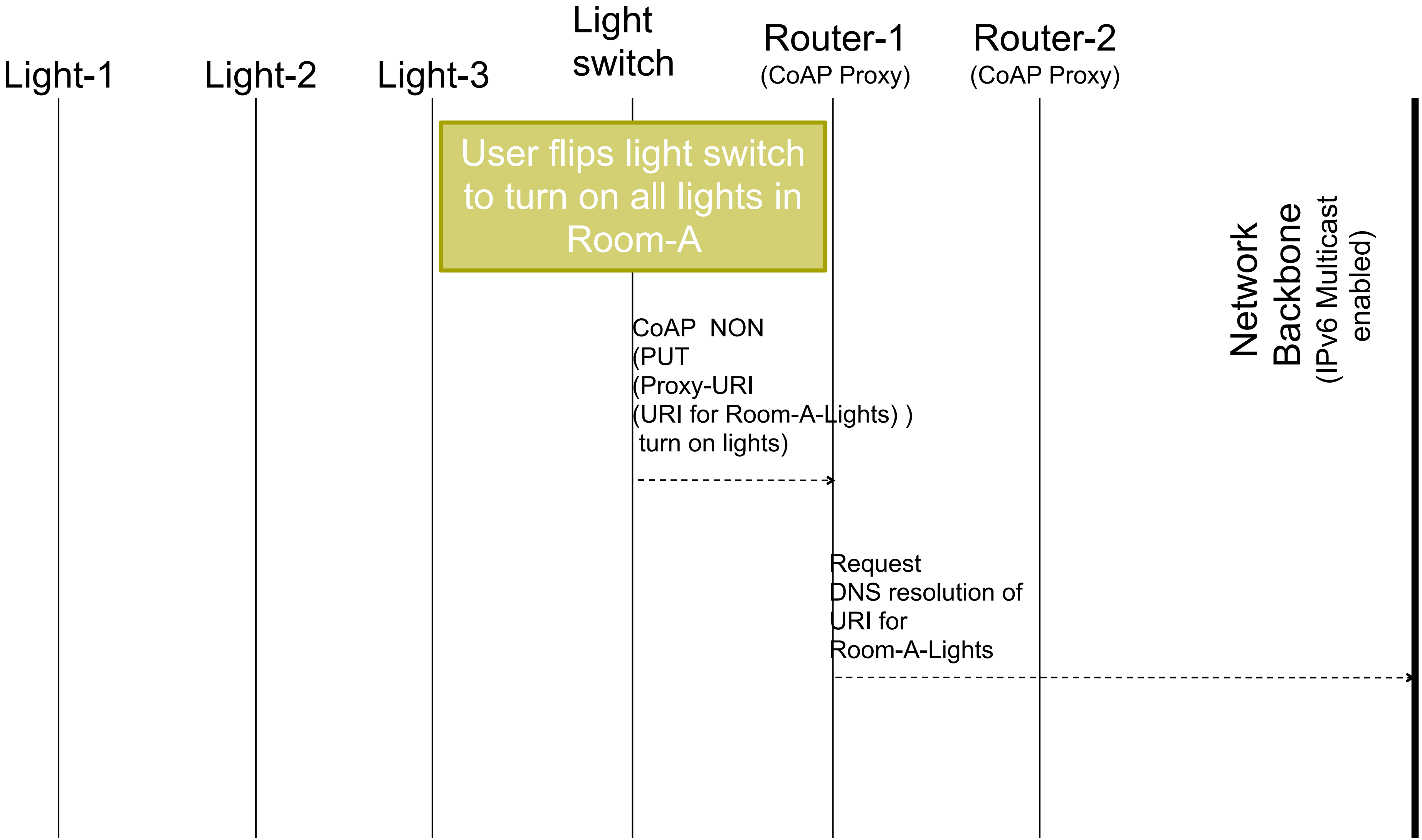
MLD Report: Join Group (Room-A-Lights)

MLD Report: Join Group (Room-A-Lights)

MLD Report: Join Group (Room-A-Lights)

MLD Report: Join Group (Room-A-Lights)

Turning on lights in Room-A (3/5)



Light-1

Light-2

Light-3

Light switch

Router-1
(CoAP Proxy)

Router-2
(CoAP Proxy)

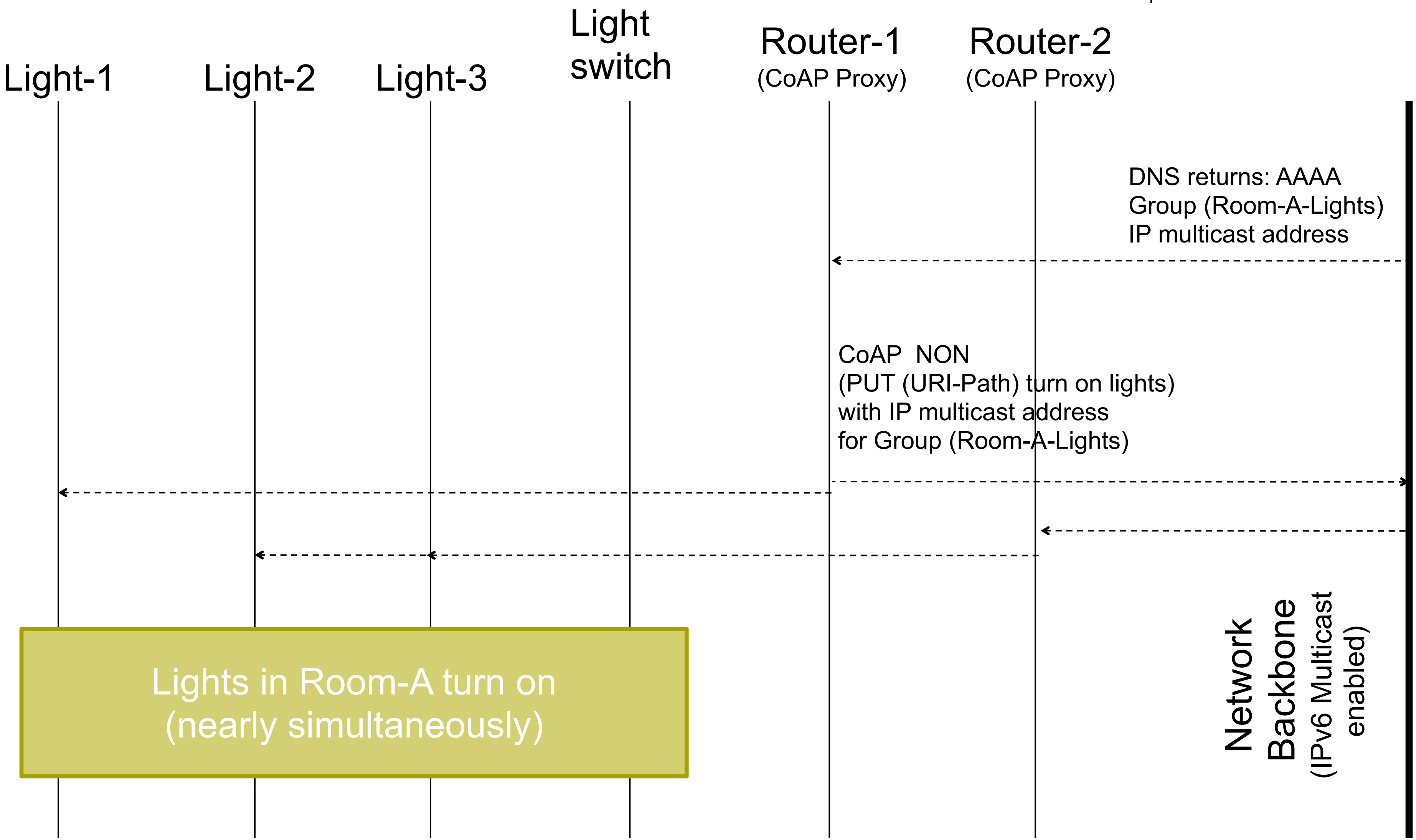
Network Backbone
(IPv6 Multicast enabled)

User flips light switch to turn on all lights in Room-A

CoAP NON
(PUT
(Proxy-URI
(URI for Room-A-Lights))
turn on lights)

Request
DNS resolution of
URI for
Room-A-Lights

Turning on lights in Room-A (4/5)



Light-1

Light-2

Light-3

Light switch

Router-1
(CoAP Proxy)

Router-2
(CoAP Proxy)

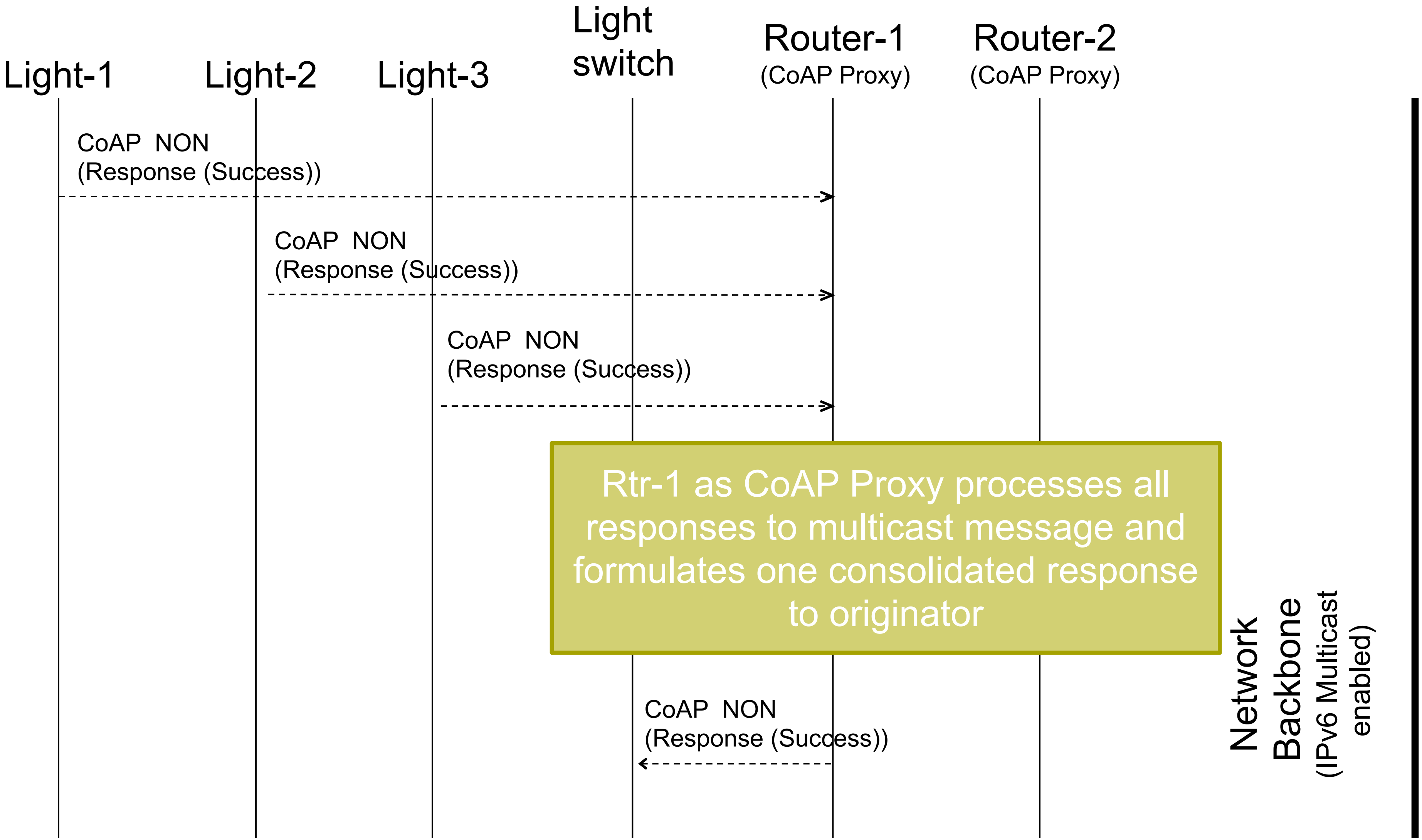
DNS returns: AAAA
Group (Room-A-Lights)
IP multicast address

CoAP NON
(PUT (URI-Path) turn on lights)
with IP multicast address
for Group (Room-A-Lights)

Lights in Room-A turn on
(nearly simultaneously)

Network
Backbone
(IPv6 Multicast
enabled)

Turning on lights in Room-A (5/5)


Light-1
Light-2
Light-3
Light switch
Router-1 (CoAP Proxy)
Router-2 (CoAP Proxy)
CoAP NON (Response (Success))
CoAP NON (Response (Success))
CoAP NON (Response (Success))
CoAP NON (Response (Success))
Network Backbone (IPv6 Multicast enabled)

Rtr-1 as CoAP Proxy processes all responses to multicast message and formulates one consolidated response to originator

SOME REMAINING OPEN ISSUES

Options for Group Resource Manipulation (1/4)

- There are several considerations for CoAP group resource manipulation:
  - IP multicast does not guarantee delivery of messages to all members of a group (i.e. can have lost IP messages)
    - And for CoAP all multicast messages must be sent as Non-Confirmable, and the server may ignore requests
  - Also IP multicast does not allow a sender to know how many members in a multicast group
  - So cannot determine directly if message was received or not by all group members. Can only send repeated multicast messages for enhanced reliability (but this obviously can cause congestion if not used carefully)

Options for Group Resource Manipulation (2/4)

- (1) Use group communications only for safe methods: GET
  - GET is a safe method (i.e. only information retrieval and should not cause state change in server for one or multiple requests) and so can be theoretically repeated
  - However, for congestion control purposes, should minimize repeat of GET multicast messages (i.e. is there a real cost if the GET was not delivered to all members?)

Options for Group Resource Manipulation (3/4)

- (2) Use group communications only for idempotent methods: GET, PUT, DELETE
  - GET is both safe and idempotent
  - PUT, DELETE are idempotent (multiple identical requests have the same effect as a single request) and so can be theoretically repeated
  - However, for congestion control purposes, should only repeat PUT, DELETE multicast messages for critical cases (i.e. is there a real cost if the PUT/DELETE was not delivered to all members?)

Options for Group Resource Manipulation (4/4)

- (3) Use group communications for all methods: GET, PUT, DELETE, POST
  - GET, PUT, DELETE are theoretically possible to repeat as multicast messages (as safe and/or idempotent) but this should be carefully minimized
  - POST is neither safe nor idempotent. Therefore, cannot repeat the POST request as a multicast message

Friday = Casual Day!

CoRE: Constrained RESTful enviro nts

The

“how many engineers
does it take
to light up a light bulb”

WG

- **We assume people have read the drafts**
- **Meetings serve to advance difficult issues by making good use of face-to-face communications**
- **Be aware of the IPR principles, according to RFC 3979 and its updates**

✓Blue sheets
✓Scribe(s)

83rd IETF: core WG Agenda

09:00	Introduction, Agenda, Status	Chairs (10)
~~09:10~~	~~SOS Ws Report~~	~~HT (10)~~
09:20	ETSI Plugtest Report	CB (10)
09:30	1 – core Link Format	ZS (90)
09:50	1 – core, block, observe, WGLC	ZS, KH, CB (20)
10:45	Interfaces	ZS (45)
11:30 retire to Friday, 09:00	Intro	Chairs (05)
09:05	SOS Ws Report for real	HT (10)
09:15	1 – core, block, observe, WGLC	(15)
09:30	Group 4 – DNA	MI PV (30)
10:00	Group 5 – “sleepy”	MV TF (30)
10:30	Group 6 – “other”	AC KL MB SL (20)
10:50	Group 7 – “next steps”	Chairs (10)
11:00 retire		

83rd IETF: core WG Agenda

09:00	Introduction, Agenda, Status	Chairs (10)
~~09:10~~	~~SOS Ws Report~~	~~HT (10)~~
09:20	ETSI Plugtest Report	CB (10)
09:30	1 – core Link Format	ZS (90)
09:50	1 – core, block, observe, WGLC	ZS, KH, CB (20)
10:45	Interfaces	ZS (45)
11:30	retire to Friday, 09:00 Intro	Chairs (05)
09:05	SOS Ws Report for real	HT (10)
09:15	1 – core, block, observe, WGLC	(15)
09:30	Group 4 – DNA	MI PV (30)
10:00	Group 5 – “sleepy”	MV TF (30)
10:30	Group 6 – “other”	AC KL MB SL (20)
10:50	Group 7 – “next steps”	Chairs (10)
11:00	retire	

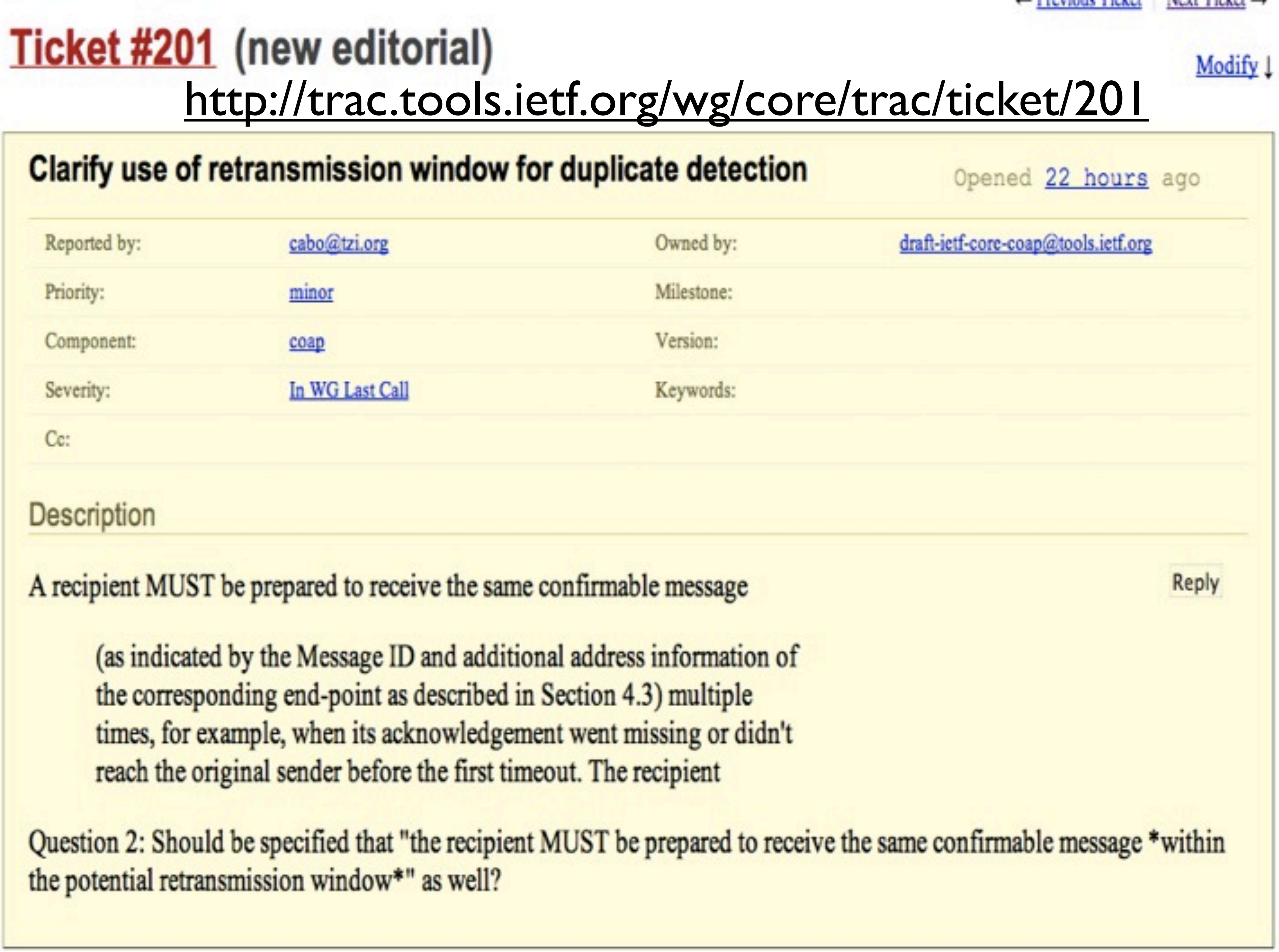
Insert SOS slides here

83rd IETF: core WG Agenda

09:00	Introduction, Agenda, Status	Chairs (10)
~~09:10~~	~~SOS Ws Report~~	~~HT (10)~~
09:20	ETSI Plugtest Report	CB (10)
09:30	1 – core Link Format	ZS (90)
09:50	1 – core, block, observe, WGLC	ZS, KH, CB (20)
10:45	Interfaces	ZS (45)
11:30	retire to Friday, 09:00 Intro	Chairs (05)
09:05	SOS Ws Report for real	HT (10)
09:15	1 – core, block, observe, WGLC	(15)
09:30	Group 4 – DNA	MI PV (30)
10:00	Group 5 – “sleepy”	MV TF (30)
10:30	Group 6 – “other”	AC KL MB SL (20)
10:50	Group 7 – “next steps”	Chairs (10)
11:00	retire	

Group 1:WGLC

coap-09, block-08, observe-05



Ticket #201 (new editorial)

http://trac.tools.ietf.org/wg/core/trac/ticket/201

Clarify use of retransmission window for duplicate detection

Opened 22 hours ago

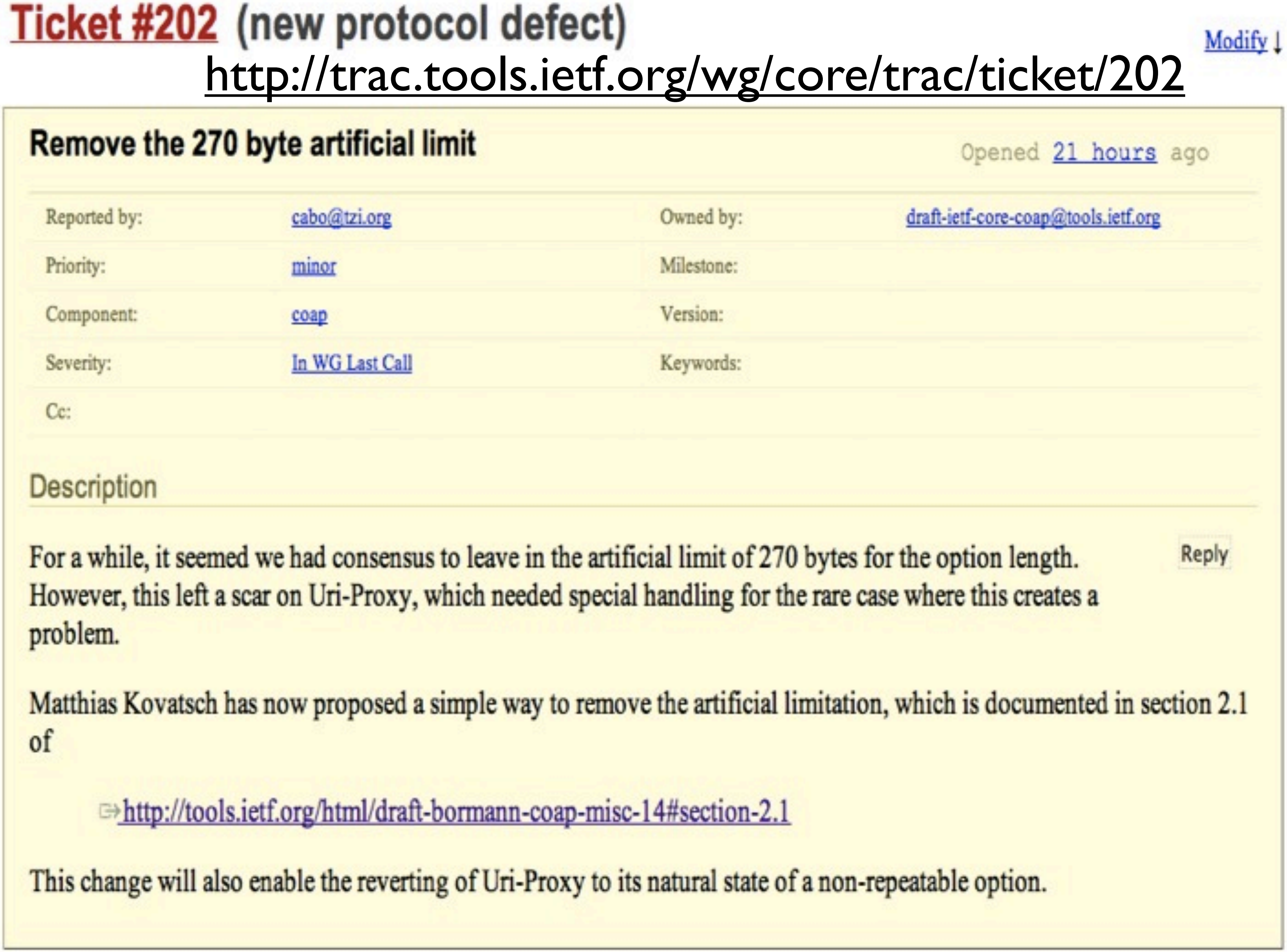
Reported by:	cabo@tzi.org	Owned by:	draft-ietf-core-coap@tools.ietf.org
Priority:	minor	Milestone:	
Component:	coap	Version:	
Severity:	In WG Last Call	Keywords:	
Cc:			

Description

A recipient MUST be prepared to receive the same confirmable message

> (as indicated by the Message ID and additional address information of the corresponding end-point as described in Section 4.3) multiple times, for example, when its acknowledgement went missing or didn't reach the original sender before the first timeout. The recipient

Question 2: Should be specified that "the recipient MUST be prepared to receive the same confirmable message *within the potential retransmission window*" as well?



Ticket #202 (new protocol defect)

http://trac.tools.ietf.org/wg/core/trac/ticket/202

Remove the 270 byte artificial limit

Opened 21 hours ago

Reported by:	cabo@tzi.org	Owned by:	draft-ietf-core-coap@tools.ietf.org
Priority:	minor	Milestone:	
Component:	coap	Version:	
Severity:	In WG Last Call	Keywords:	
Cc:			

Description

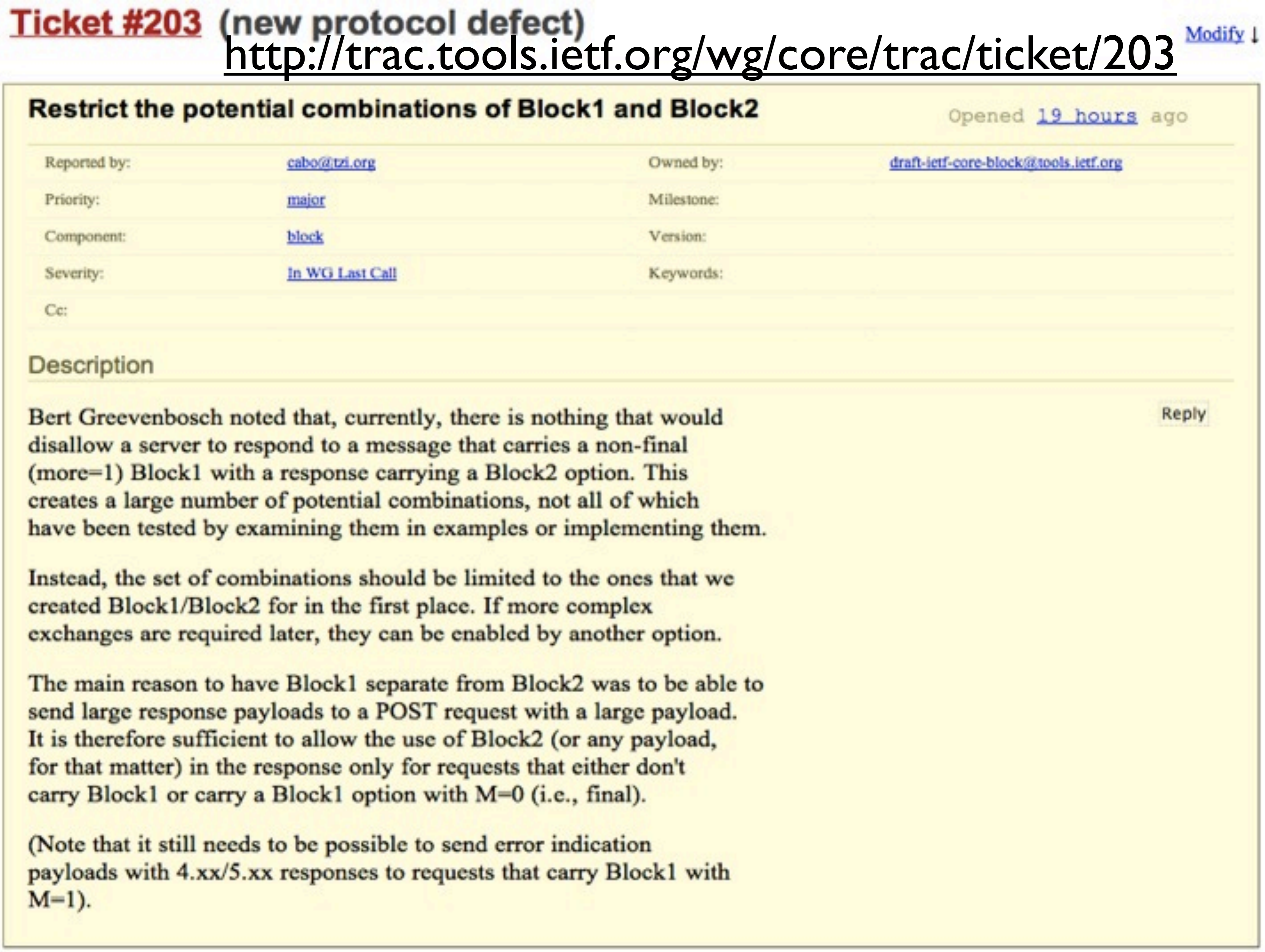
For a while, it seemed we had consensus to leave in the artificial limit of 270 bytes for the option length. However, this left a scar on Uri-Proxy, which needed special handling for the rare case where this creates a problem.

Matthias Kovatsch has now proposed a simple way to remove the artificial limitation, which is documented in section 2.1 of

⇒http://tools.ietf.org/html/draft-bormann-coap-misc-14#section-2.1

This change will also enable the reverting of Uri-Proxy to its natural state of a non-repeatable option.

http://trac.tools.ietf.org/wg/core/trac/ticket/203



Ticket #203 (new protocol defect)

Restrict the potential combinations of Block1 and Block2

Opened 19 hours ago

Reported by:	cabo@tzi.org	Owned by:	draft-ietf-core-block@tools.ietf.org
Priority:	major	Milestone:	
Component:	block	Version:	
Severity:	In WG Last Call	Keywords:	
Cc:			

Description

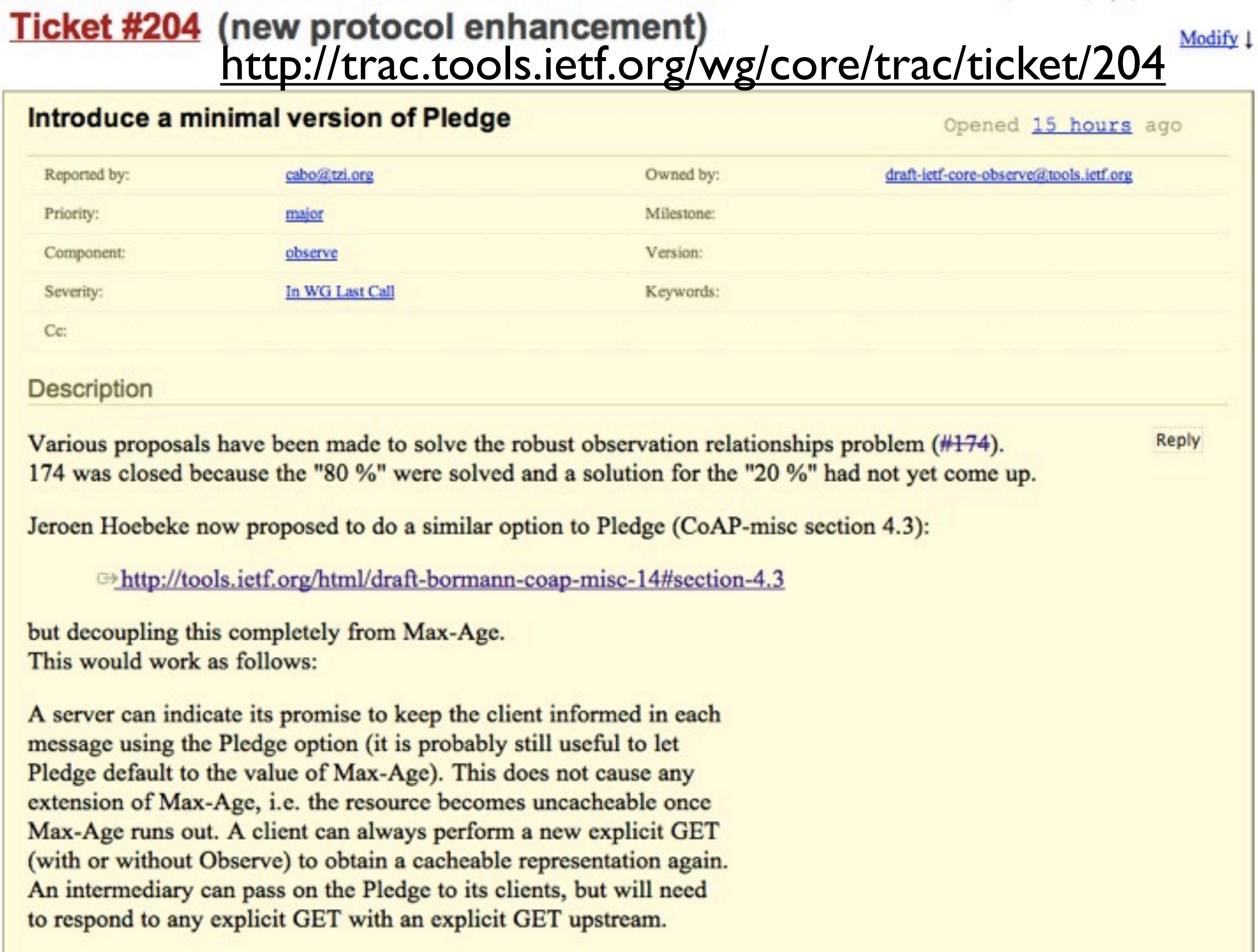
Bert Greevenbosch noted that, currently, there is nothing that would disallow a server to respond to a message that carries a non-final (more=1) Block1 with a response carrying a Block2 option. This creates a large number of potential combinations, not all of which have been tested by examining them in examples or implementing them.

Instead, the set of combinations should be limited to the ones that we created Block1/Block2 for in the first place. If more complex exchanges are required later, they can be enabled by another option.

The main reason to have Block1 separate from Block2 was to be able to send large response payloads to a POST request with a large payload. It is therefore sufficient to allow the use of Block2 (or any payload, for that matter) in the response only for requests that either don't carry Block1 or carry a Block1 option with M=0 (i.e., final).

(Note that it still needs to be possible to send error indication payloads with 4.xx/5.xx responses to requests that carry Block1 with M=1).

http://trac.tools.ietf.org/wg/core/trac/ticket/204



Ticket #204 (new protocol enhancement)

Introduce a minimal version of Pledge

Opened 15 hours ago

Reported by:	cabo@tzi.org	Owned by:	draft-ietf-core-observe@tools.ietf.org
Priority:	major	Milestone:	
Component:	observe	Version:	
Severity:	In WG Last Call	Keywords:	
Cc:			

Description

Various proposals have been made to solve the robust observation relationships problem (~~#174~~).
174 was closed because the "80 %" were solved and a solution for the "20 %" had not yet come up.

Jeroen Hoebeke now proposed to do a similar option to Pledge (CoAP-misc section 4.3):

http://tools.ietf.org/html/draft-bormann-coap-misc-14#section-4.3

but decoupling this completely from Max-Age.
This would work as follows:

A server can indicate its promise to keep the client informed in each message using the Pledge option (it is probably still useful to let Pledge default to the value of Max-Age). This does not cause any extension of Max-Age, i.e. the resource becomes uncacheable once Max-Age runs out. A client can always perform a new explicit GET (with or without Observe) to obtain a cacheable representation again. An intermediary can pass on the Pledge to its clients, but will need to respond to any explicit GET with an explicit GET upstream.

83rd IETF: core WG Agenda

09:00	Introduction, Agenda, Status	Chairs (10)
~~09:10~~	~~SOS Ws Report~~	~~HT (10)~~
09:20	ETSI Plugtest Report	CB (10)
09:30	1 – core Link Format	ZS (90)
09:50	1 – core, block, observe, WGLC	ZS, KH, CB (20)
10:45	Interfaces	ZS (45)
11:30	retire to Friday, 09:00 Intro	Chairs (05)
09:05	SOS Ws Report for real	HT (10)
09:15	1 – core, block, observe, WGLC	(15)
09:30	Group 4 – DNA	MI PV (30)
10:00	Group 5 – “sleepy”	MV TF (30)
10:30	Group 6 – “other”	AC KL MB SL (20)
10:50	Group 7 – “next steps”	Chairs (10)
11:00	retire	

Group 4: Discovery, Naming, Addressing

Constrained Application Autoconfiguration

draft-nieminen-core-service-discovery-02

IETF 83

markus.isomaki@nokia.com

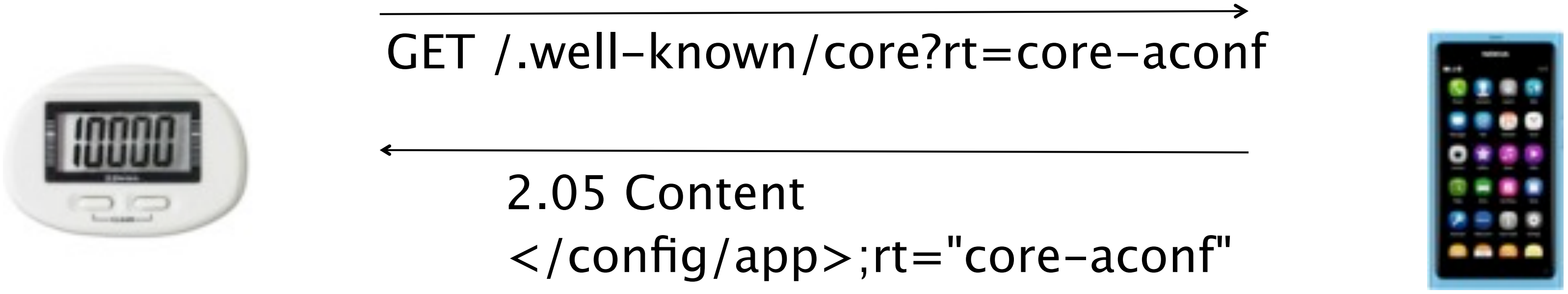
Constrained Application Autoconfiguration

- Many small or UI-less devices are not directly configurable
- CoAP client needs some basic parameters to bootstrap itself
  - Server URL or address
  - Credentials (optionally)
- Users will have to set these somehow

- Options
  1. Run a small web server on the device
     - May be infeasible in some extreme situations where CoAP is used
  2. Hard code an address for a configuration website
     - Dependency on the device provider
  3. Some kind of configuration discovery
     - **Focus of this draft, based on CoAP**

CoAP based configuration discovery

- Assumptions
  - The constrained device supports CoAP
  - User has a CoAP “configuration server” in her local domain

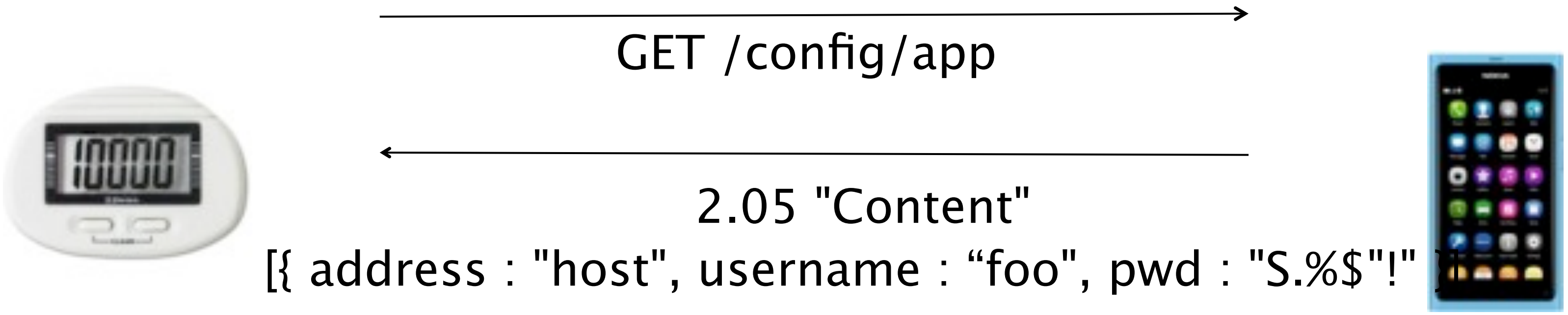


GET /.well-known/core?rt=core-aconf

2.05 Content
</config/app>;rt="core-aconf"

- Discovery request is sent to a well-known multicast address
- Can include device ID and service type to identify type of configuration
- User has either manually entered the config on the server or gotten it that far by some other means (HTTP GET via browser/ app, OMA DM, SMS, …)

Configuration fetching and data model



GET /config/app

2.05 "Content"
[{ address : "host", username : “foo", pwd : "S.%$"!" }]

- Standardized configuration to contain only the minimum set of mandatory parameters to bootstrap the device
- After getting the config the device is good to go and do whatever the application is supposed to do

Security

- Configuration client and server must authenticate or at least authorize each other
- Configuration transfer must be confidentiality and integrity protected

- Two main options for CoAP and low-power radio devices
  - DTLS
    - Shared key, public keys, certs?
  - Layer 2 security between devices (e.g. based on BT-LE)
    - Pairing mechanisms exist
    - Channel biding from CoAP to L2 security?
- This makes it challenging but user friendly options for bootstrapping security between two devices owned and operated by the same user should be possible
  - PIN codes, public key in the sales box, …
  - New work item by itself?

Points of Discussion

- Is this a problem IETF should solve?
- Should we also think what configuration a CoAP **server** needs?
- Is this approach worth pursuing?
- Is CoAP the right protocol?
- Is there a feasible way to bootstrap security?

- Take this work into CoRE WG?

CoAP Utilization for Building Control

draft-vanderstok-core-dna-01

Discovery, Naming, Addressing

Peter van der Stok
Kerry Lynn
Anders Brandt

March 30,2012

Service Discovery, why

Standards Developing Organizations provide standards
to name services and their attributes for ecosystems of companies

Device names are project dependent

Necessary to find device names – with ports, Function sets,…..
from agreed and advertized service names

What Objects Must We Name/Resolve/Discover?

Device	Physical object bound to at least one IP address (A, AAAA) and optionally a UID
Multicast Group	A set of devices listening on a common multicast address
End-point	{protocol, host, port} tuple, where *host* may name a (device or group) and *port* may have a default value
Function set	Service type, end-point (SRV) and path (TXT)
Function set Collection	Parent path for a set of subordinate Function sets

Service Discovery, device

Find devices with given service (at given location)

E.g.: Presence sensor needs all lamps in given office

lamp1
lamp2
myoffice
pir

E.g.: pir.myoffice.acme.com looks for devices
in domain myoffice.acme.com
with service: _onoff._sub._bc._udp

Possible solution in DNS-SD:

PTR Resource Records identify Function sets

```
_onoff._sub._bc.udp.myoffice.acme.com   PTR   light1.myoffice.acme.com
_onoff._sub._bc.udp.myoffice.acme.com   PTR   light2.myoffice.acme.com
```

PTR RRs returned to PIR with names of SRVs and TXT RRs
SRV provides host name, port; where host name resolves to IP address.

Service Discovery, group

Grouping of devices according to domains is often sufficient
but in general too restrictive

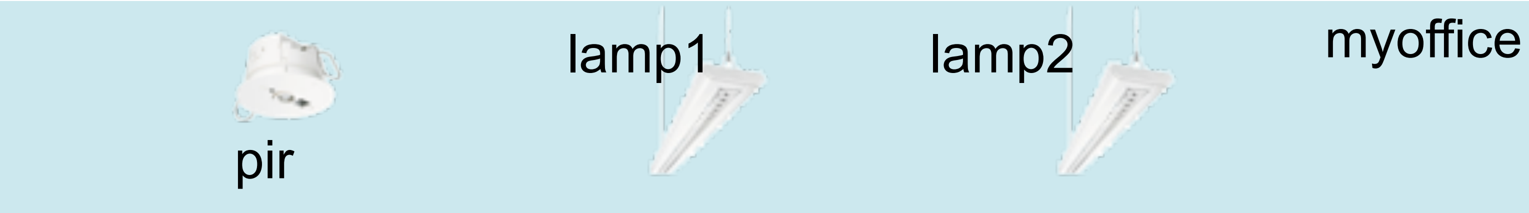
In the home, no domains may exist,
but clear separation between rooms is evident

In office for example, different settings for window part of offices

Devices need to be grouped independent of domain as well,
possibly in conjunction with multicast group

Service Discovery, group

Device multicasts to group of devices



E.g.: pir.myoffice.example.com multicasts to group in domain acme.com
with service: _onoff._sub._bc._udp

Possible example solution:

Use remote control or commissioning device:
to define Group with PIR and Lamp1, Lamp2

PIR and Lamps query to which groups they belong

Group names are returned, Multicast IP address is found

- PIR uses IP address to send data
- Lamp enables reception of IP address.

Service Discovery, group

Device multicasts to group of devices

E.g.: pir.myoffice.example.com multicasts to group in domain acme.com

Possible Implementation with DNS-SD:

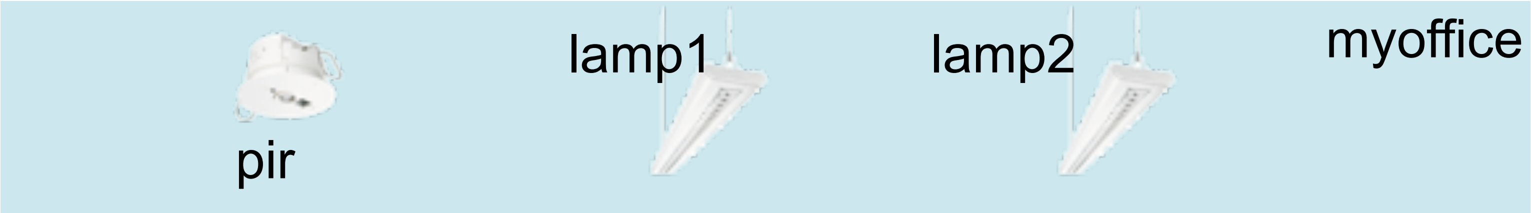
lamp1.acme.com	PTR	MyGroup.acme.com
lamp2.acme.com	PTR	MyGroup.acme.com
pir.acme.com	PTR	MyGroup.acme.com

MyGroup must be identified as group-name by SDO

Mygroup.acme.com	AAAA	ff15::16
6.1.0.0……5.1.f.f	PTR	Mygroup.acme.com
6.1.0.0……5.1.f.f	PTR	lamp1.acme.com
6.1.0.0……5.1.f.f	PTR	lamp2.acme.com
6.1.0.0……5.1.f.f	PTR	pir.acme.com

Service Discovery, group

Device multicasts to group of devices



E.g.: pir.myoffice.example.com multicasts to group in domain acme.com

Example implementation shows:

Given group name, find group members via reverse address resolution
extension of existing reverse address resolution to multicast address
Given device name, find associated group
extension of PTR functionality for groups

83rd IETF: core WG Agenda

09:00	Introduction, Agenda, Status	Chairs (10)
~~09:10~~	~~SOS Ws Report~~	~~HT (10)~~
09:20	ETSI Plugtest Report	CB (10)
09:30	1 – core Link Format	ZS (90)
09:50	1 – core, block, observe, WGLC	ZS, KH, CB (20)
10:45	Interfaces	ZS (45)
11:30	retire to Friday, 09:00 Intro	Chairs (05)
09:05	SOS Ws Report for real	HT (10)
09:15	1 – core, block, observe, WGLC	(15)
09:30	Group 4 – DNA	MI PV (30)
10:00	Group 5 – “sleepy”	MV TF (30)
10:30	Group 6 – “other”	AC KL MB SL (20)
10:50	Group 7 – “next steps”	Chairs (10)
11:00	retire	

Group 5:“sleepy”

CoRE objectives

- Charter

> A constrained IP network has limited packet sizes, may exhibit a high degree of packet loss, and may **have a substantial number of devices that may be powered off at any point in time but periodically "wake up" for brief periods of time.**

> CoAP will support various forms of "caching". For example, if a temperature sensor **is normally asleep but wakes up every five minutes** and sends the current temperature to a proxy that has subscribed, when the proxy receives a request over HTTP for that temperature resource, it can respond with the last seen value instead of trying to query the Device which is currently asleep.

- draft-shelby-core-coap-req

> REQ3: **The ability to deal with sleeping nodes.** Devices may be powered off at any point in time but periodically "wake up" for brief periods of time.

Why sleep?

- Save batteries
  - Expect 5-10 year lifetime
- Improve energy balance for harvester
  - Improve operation time in unfavorable conditions (dark…)
  - Harvester: Photovoltaic cell, Current Transformer
- Sporadic energy production
  - No energy storage, unpredictable availability
  - Harvester: Mechanical stress, Temperature difference

Sleeping modes

- Link sleep state : radio duty cycle
  - Off < 1s
  - Always on illusion
  - Server model still applicable
- Link disconnected state : radio off
  - Off > 1s, typical 10min
  - **Server model not working**
  - Need to store state on another web entity

Current state

- draft-arkko-core-sleepy-sensors conclusions
- Server model
  - Basic CoAP usage
  - Requires always on link
- Observer model
  - Efficient operation mode
  - Registration requires server model
- Client model
  - Most efficient
  - Not well defined

Conclusion

- Need to define new architecture for sleeping devices
- CoRE is going to miss the range of
  - Energy harvesting sensors
  - Battery-operated sensors with long lifetime
- CoRE should provide a standard method to support sleeping devices

Goals

- A sleeping endpoint (SEP) is client-only
- SEP delegates resource hosting to proxy but keeps ownership
- Resources available when SEP is asleep
- Resource discovery with semantic
- Auto configuration: SEP can discover proxy
- Application profile agnostic mechanism
- REST design for base functions
- CoAP extensions for optimizations

Concepts

- Reverse proxy
  - Map resources in the proxy resource tree
- Caching proxy
  - Store copy, serve content while SEP asleep
- link-format
  - Detailed resource description
  - Multicast Discovery
- RD interface
  - Same interface but MP is not RD
  - Shared REST design
    - Facilitate export to RD
    - Optimizations when RD and MP are collocated

Mirror Proxy discovery

- Reuse standard resource discovery

```
SEP                                                MP
 |                                                 |
 | ----- GET /.well-known/core?rt=core-mp ------>  |
 |                                                 |
 |                                                 |
 | <---- 2.05 Content "</mp>; rt="core-mp" ------  |
 |                                                 |

Req: GET coap://[ff02::1]/.well-known/core?rt=core-mp
Res: 2.05 Content
</mp>;rt="core-mp"
```

Registration

- Reuse RD interface as is but change actions
- Link-format description like ordinary web server
- SEP’s resources => Sub-resources of MP entry

```
SEP                                              MP
   |                                               |
   | --- POST /mp "</dev..." ------------->        |
   |                                               |
   |                                               |
   | <-- 2.01 Created Location: /mp/0 ------       |
   |                                               |
```

```
Req: POST coap://mp.example.org/mp?
       h=switch4602
Etag: 0x3f
Payload:
</dev/>;rt="ipso:dev",
</dev/mfg >;rt="ipso:dev-mfg",
</dev/mdl>;rt="ipso:dev-mdl",
</dev/n>;rt="ipso:dev-name",
</lt/>;rt="ipso:lt",
</lt/ctr>;rt="ipso:lt-ctr"

Res: 2.01 Created
Location: /mp/0
```

Resource tree

- Server endpoint on MP node can hosts its own function sets

```
</dev/>;rt="ipso:dev",
</dev/mfg >;rt="ipso:dev-mfg",
</dev/mdl>;rt="ipso:dev-mdl",
</dev/n>;rt="ipso:dev-name",
</pwr/>;rt="ipso:pwr",
</pwr/kwh>;rt="ipso:pwr-kwh“
</mp/>;rt=“core-mp”,
```

Node’s own resources:
MP + power meter

```
</mp/0>;h="switch4602",
</mp/0/dev/>;rt="ipso:dev",
</mp/0/dev/mfg >;rt="ipso:dev-mfg",
</mp/0/dev/mdl>;rt="ipso:dev-mdl",
</mp/0/dev/n>;rt="ipso:dev-name",
</mp/0/lt/>;rt="ipso:lt",
</mp/0/lt/ctr>;rt="ipso:lt-ctr"
```

Mirrored resources:
Virtual light switch

Resource discovery for clients

- Resources hosted on MP
  - Clients get well-known/core on MP
- MP export to RD if present
  - Separate RD entry

- Example of RD export

```
MP                                                  RD
    |                                                |
    | --- POST /rd "</mp/0..." ------------>         |
    |                                                |
    |                                                |
    | <-- 2.01 Created Location: /rd/6534 ---        |
    |                                                |
```

```
Req: POST coap://rd.example.org/rd?
      h=switch4602
Etag: 0x6a
Payload:
</mp/0/dev/>;rt="ipso:dev",
</mp/0/dev/mfg >;rt="ipso:dev-mfg",...


Res: 2.01 Created
Location: /rd/6534
```

Update mirrored resource

- Simple PUT requests
- MP can’t contact SEP
  - Current caching model not applicable
  - Max-Age: fine-grained lifetime management
- Observation available

```
SEP                                           MP
    |                                          |
    | --- PUT /mp/0/lt/ctr "1" ----------->    |
    |                                          |
    |                                          |
    | <-- 2.04 Changed ---------------------   |
    |                                          |

   Req: PUT coap://mp.example.org/mp/0/lt/ctr
          (Max-Age: 1 hour)
   Payload: 1
   Res: 2.04 Changed
```

Writable resources

- Triggering rules to refresh resources
  - Polling: Energy saving entails long polling intervals
  - HMI: Press button to update settings
- What-has-changed model
  - New interface with list of links for updated resources

```
SEP                          MP                       Client
  |                           |                          |
  | - PUT /mp/0/dev/n ------> |                          |
  |                           |                          |
  | <- 2.04 Changed --------- |                          |
  |                           |                          |
  |                           | <-- PUT /mp/0/dev/n ---  |
  |                           |                          |
  |                           | -- 2.04 Changed ------>  |
  |                           |                          |
  | [press button]            |                          |
  |                           |                          |
  | - GET /mp/0/dev/n ------> |                          |
  |                           |                          |
  | <- 2.05 Content --------- |                          |
  |                           |                          |
```

Conclusion

- Feedback from Jari Arkko
  - Work on sleepy devices is needed
  - Overall architecture of MP is easy to understand
  - MP is easy to implement
  - RD and MP interfaces need clarifications
- Open questions
  - Security : Access control, authentication
  - Identify multiple mirrored resources from the same SEP
  - Some energy harvesting devices can’t maintain soft state on MP

Publish Option and Sleepy Nodes

T. Fossati, P. Giacomin, S. Loreto, M. Rossini

IETF 83, Paris

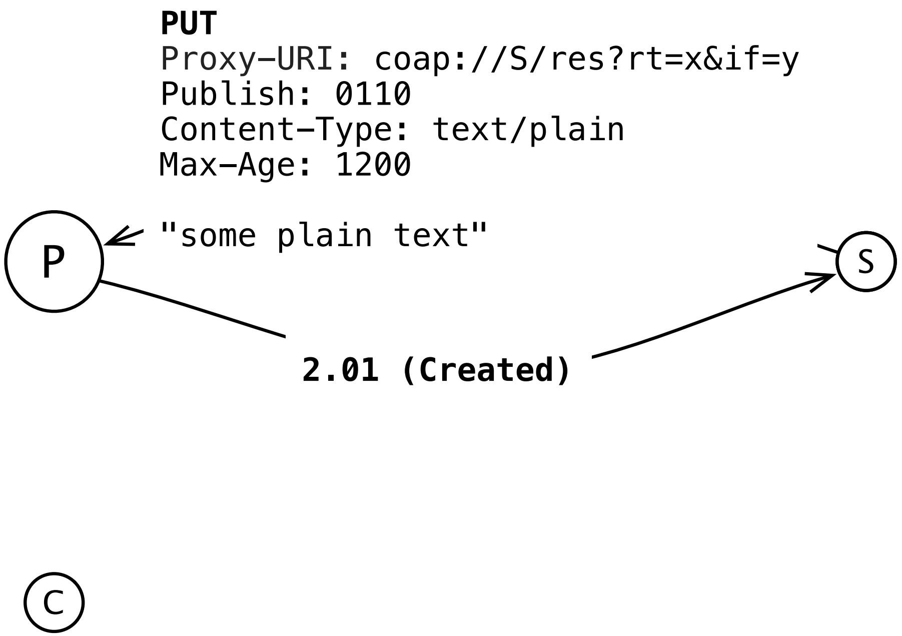
Publish

- Extend caching Proxies capabilities to help Sleepy nodes to participate in CoAP networks
- Define a simple and *native* interface for *full* delegation of resources (data, metadata and access control)

Publish (cont)

- Handle the whole delegation life-time:
  Publish → Update → ... → Update → [implicit] Remove
- Delegated Proxy may be any type:
  - Forward (reuse the published namespace)
  - Reverse (create its own namespace, a la MP)

Publish resource

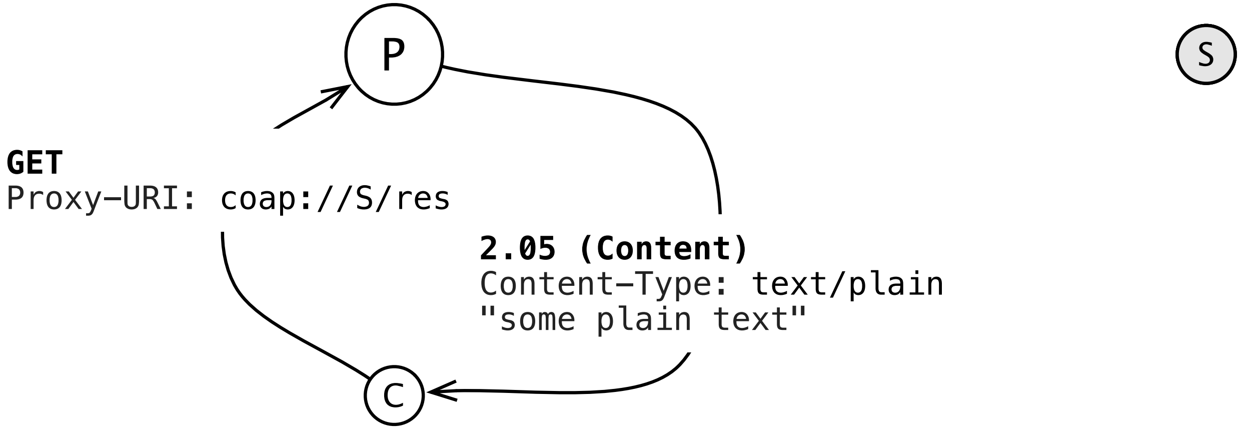


```
PUT
Proxy-URI: coap://S/res?rt=x&if=y
Publish: 0110
Content-Type: text/plain
Max-Age: 1200

"some plain text"
```

2.01 (Created)

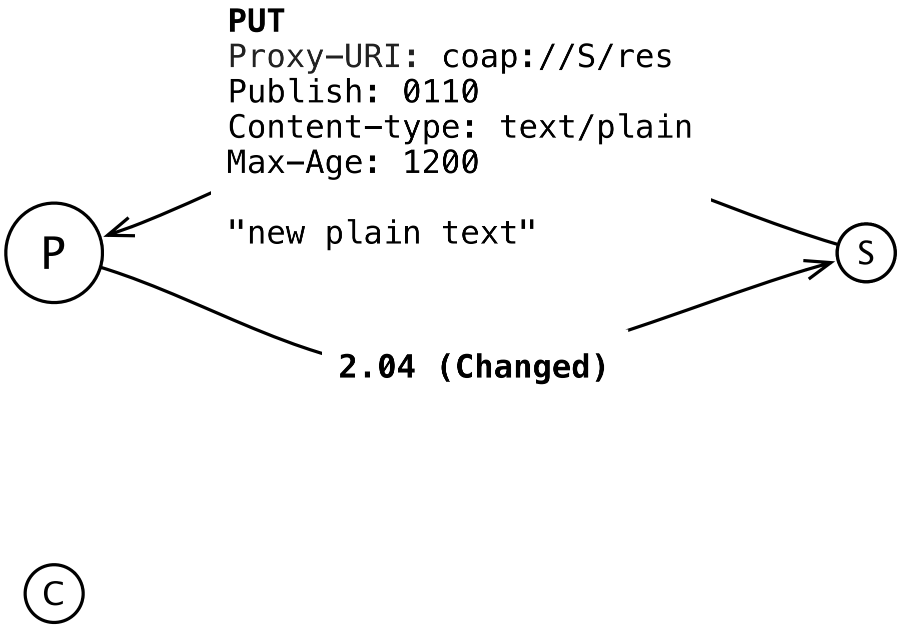
Act on published resource (forward scenario)



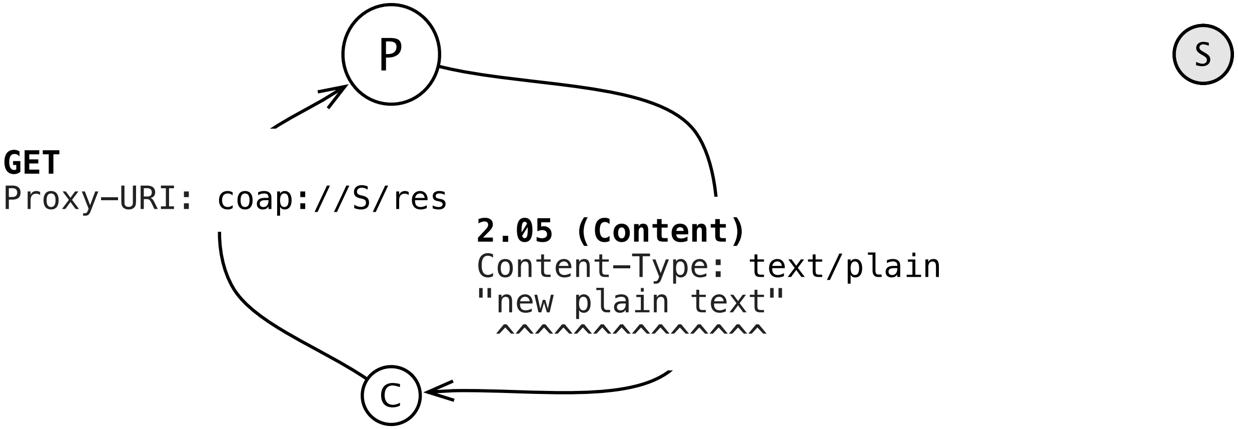
GET
Proxy-URI: coap://S/res

2.05 (Content)
Content-Type: text/plain
"some plain text"

Update published resource



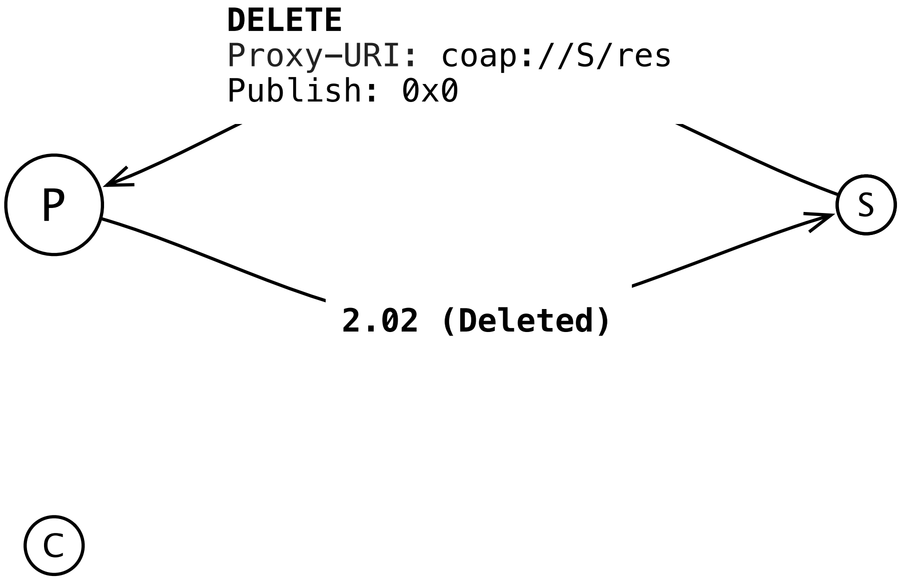
Act on updated resource (forward scenario)



```
GET
Proxy-URI: coap://S/res
```

```
2.05 (Content)
Content-Type: text/plain
"new plain text"
 ^^^^^^^^^^^^^^
```

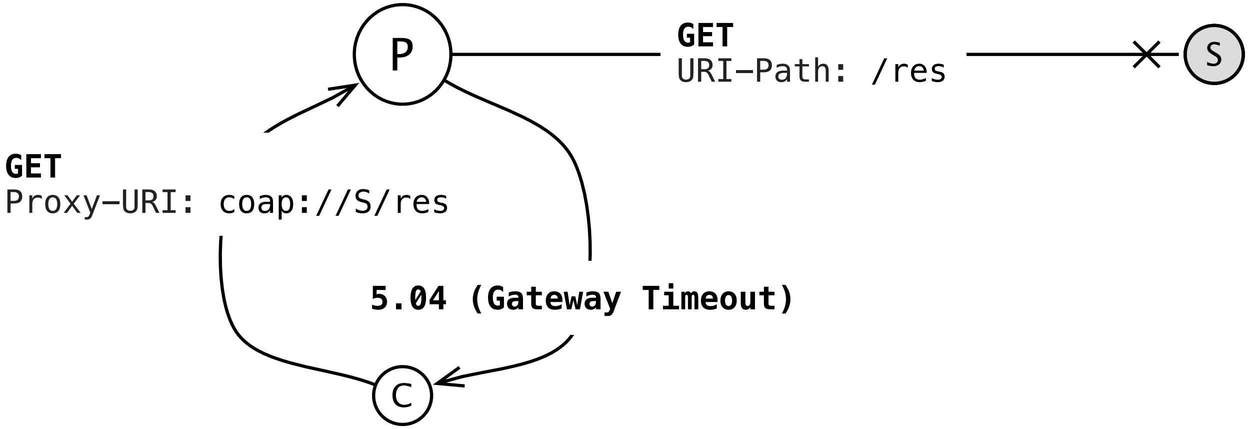
Remove delegation



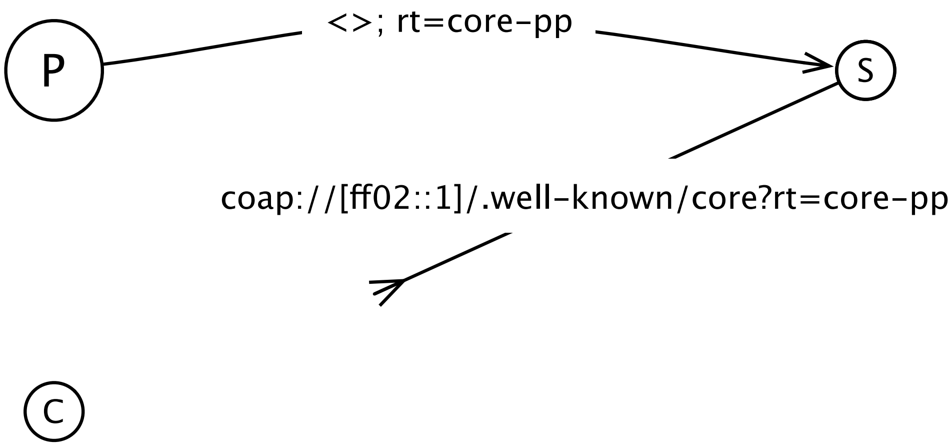
DELETE
Proxy-URI: coap://S/res
Publish: 0x0

2.02 (Deleted)

Effect of delegation removal (forward scenario)



Discover the Proxy



P

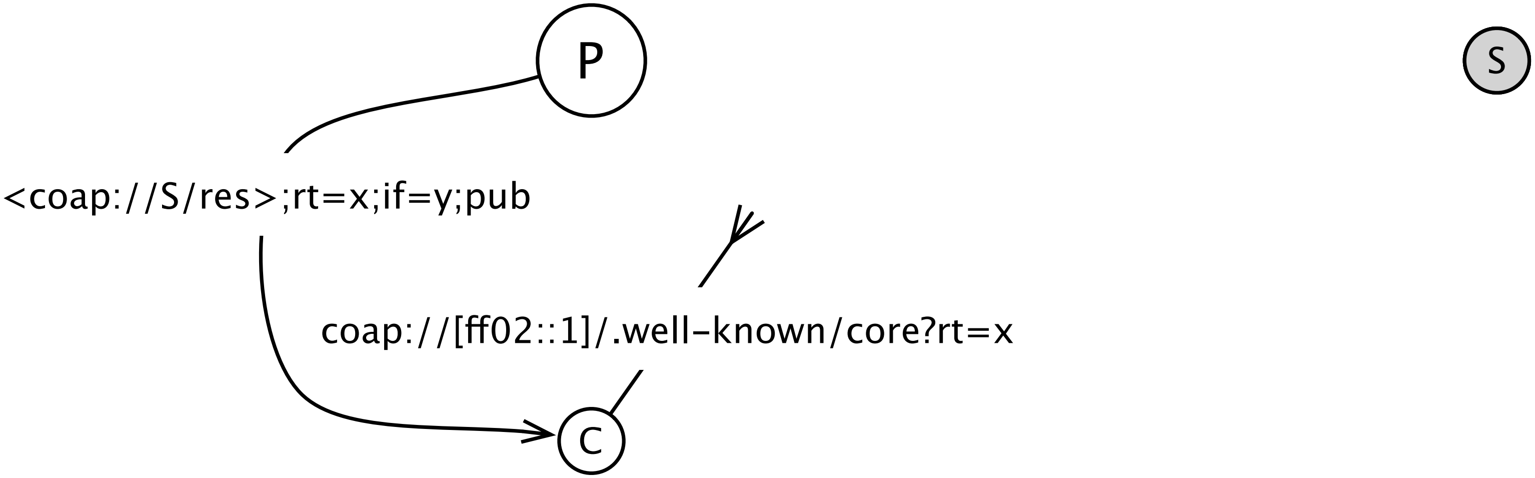
<>; rt=core-pp

S

coap://[ff02::1]/.well-known/core?rt=core-pp

C

Discover the resource (forward scenario)



P
S
<coap://S/res>;rt=x;if=y;pub
coap://[ff02::1]/.well-known/core?rt=x
C

The Good

- No state has to be maintained on the Sleepy client
- The Sleepy node may decide to never listen on the radio
- Proxy model agnostic
- Delegation of data and meta is *atomic*

The Good (cont)

- Support writable resources through methods' mask
- Support Observe of Sleepy resources
- Explicit delegation lifetime
- Trivial patch to the caching Proxy logics

The Good (cont)

```
$ git diff
@@ -249,6 +251,46 @@ ec_cbrc_t proxy_req(ec_server_t *srv, void *u0, struct timeval *u1, bool u2)
+    /* Catch Publish requests. */
+    if (m == EC_COAP_PUT && ec_request_get_publish(srv, &mask) == 0)
+    {
+        ec_mt_t mt;
+        size_t pload_sz;
+        uint32_t max_age;
+
+        if (ec_request_get_max_age(srv, &max_age))
+            max_age = 3600;
+
+        if (ec_request_get_content_type(srv, &mt))
+            mt = EC_MT_TEXT_PLAIN;
+
+        const uint8_t *pload = ec_request_get_payload(srv, &pload_sz);
+
+        dbg_err_if ((res = ec_resource_new(uri, mask, max_age)) == NULL);
+        dbg_err_if (ec_resource_add_rep(res, pload, pload_sz, mt, NULL));
+        dbg_err_if (ec_filesys_put_resource(g_ctx.cache, res));
+        res = NULL;
+
+        dbg_err_if (ec_register_cb(g_ctx.coap, uri, cache_serve, NULL));
+
+        dbg_if (ec_response_set_code(srv, EC_CREATED));
+        return EC_CBRC_READY;
+    }
```

The Bad

- Can't go through Proxies (necessarily stops at first Proxy)

Open issues

- Need *strong mutual authentication* to authorize the delegation (and subsequent ops – i.e. updates and deletion) on the resource
- How key material is supposed to be exported in case the published resource is `coaps` ?

Architecture

Mirror proxy

- REST design
  - Leverage existing work (RD)
  - Support both CoAP and HTTP
  - No discovery mechanism for optional protocol features
  - Stack vendors will not implement all CoAP options

Publish option

- Protocol feature
  - Leverage on caching proxy capabilities
  - If accepted will allow seamless integration of Sleepy into CoAP networks
  - In stack function

Resource Delegation

Mirror proxy

- Registration (POST) and update (PUT) are separate steps
  - Expect long term delegation
  - All resources registered at once => virtual device
  - Easier to add metadata
  - Drawback: Registration too complex for extremely constrained devices

Publish option

- Registration + update (PUT) merged
  - Atomic operation => consistency
  - Registration == Update => Stateless for Sleepy
  - Sleepy may not listen on the radio

Kind of proxy

Mirror proxy

- Reverse proxy
- Pros
  - No client configuration
  - Seamless integration with current RD interfaces
- Cons
  - Loose Sleepy authority

Publish option

- Forward proxy
  - Same resource identifier => allows server and client mode
- Reverse (CoAP or HTTP may be used too but is currently not described)

Packet size for updates

Mirror proxy

- Scheme + Authority inferred

```
Req: PUT
Uri-Path: mp
Uri-Path: 0
Uri-Path: res
Max-Age: 3600
Content-type: text/plain
Body: 1
```

Option length: 13 (Uid + metadata sent once during registration)

Publish option

- Explicit Scheme + Authority

```
Req: PUT
Proxy-Uri: coap://switch4602/res
Publish: 0110
Max-Age: 3600
Content-type: text/plain
Body: 1
```

Option length: 27 + metadata

Sleepy state

Mirror proxy

- Proxy address
- Mirror proxy entry
  - Low overhead

Publish option

- Proxy address

So what is the best way to model this?

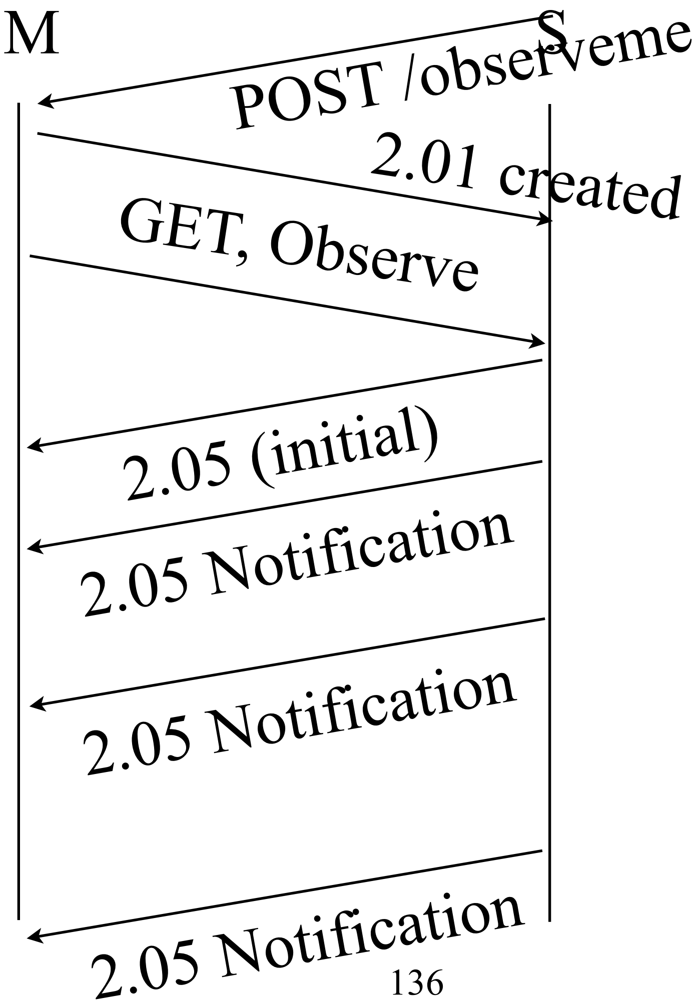
- Mirror is a special kind of proxy
- Data flows from server to mirror in a way that is controlled by server
- Probably want a CON-type message exchange
- Need to identify which resource is being updated

- So why isn’t that an Observe?

Can be done with just CoAP-09

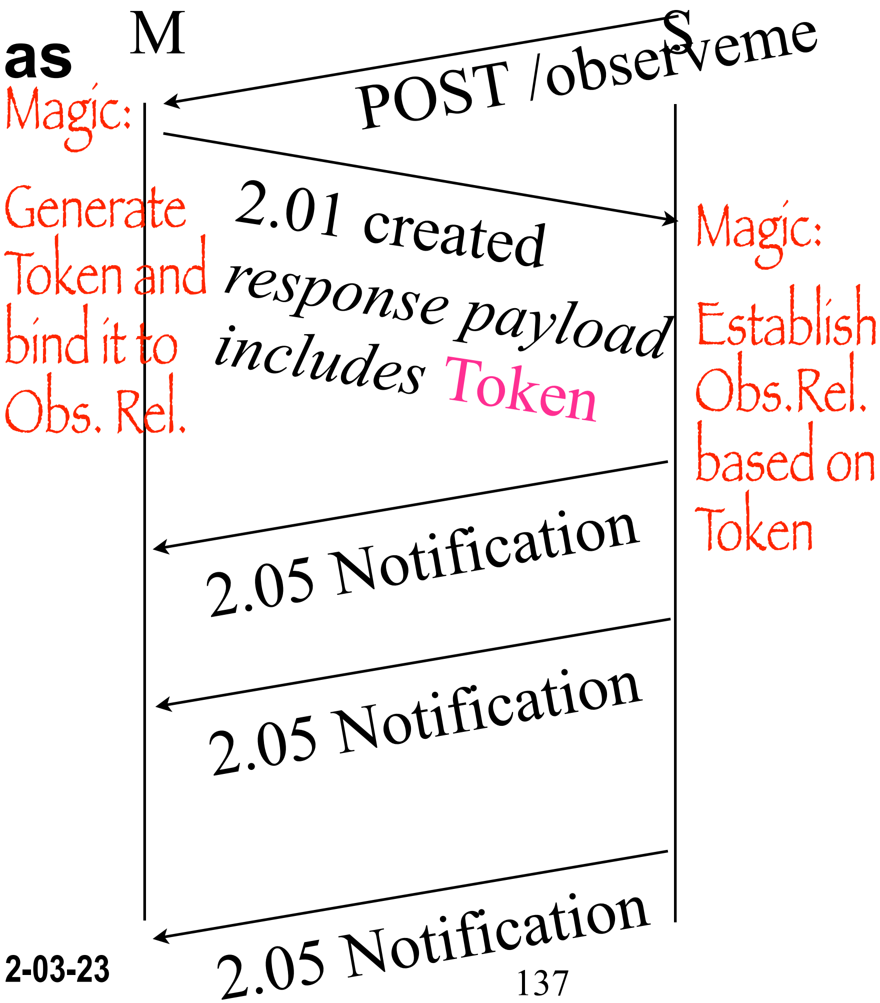
Modelling mirroring as observe

- Model the request to mirror as a POST
  - Uri actually to be mirrored
- Establish an Observation Relationship
  - Mirror supplies token to use
- Mirror then copies over representation on each notification
  - like any other proxy
  - might use CON or NON



Slight optimization

- **Model the request to mirror as a POST**
  - **Uri actually to be mirrored**
- **Establish an Observation Relationship**
  - **Mirror supplies token to use**
- **Mirror then copies over representation on each notification**
  - **like any other proxy**
  - **might use CON or NON**


M
S
POST /observeme
Magic:
Generate Token and bind it to Obs. Rel.
2.01 created
response payload includes Token
Magic:
Establish Obs.Rel. based on Token
2.05 Notification
2.05 Notification
2.05 Notification

83rd IETF: core WG Agenda

09:00	Introduction, Agenda, Status	Chairs (10)
~~09:10~~	~~SOS Ws Report~~	~~HT (10)~~
09:20	ETSI Plugtest Report	CB (10)
09:30	1 – core Link Format	ZS (90)
09:50	1 – core, block, observe, WGLC	ZS, KH, CB (20)
10:45	Interfaces	ZS (45)
11:30	retire to Friday, 09:00 Intro	Chairs (05)
09:05	SOS Ws Report for real	HT (10)
09:15	1 – core, block, observe, WGLC	(15)
09:30	Group 4 – DNA	MI PV (30)
10:00	Group 5 – “sleepy”	MV TF (30)
10:30	Group 6 – “other”	AC KL MB SL (20)
10:50	Group 7 – “next steps”	Chairs (10)
11:00	retire	

Group 6:“other”

Best practices for HTTP-CoAP mapping implementation

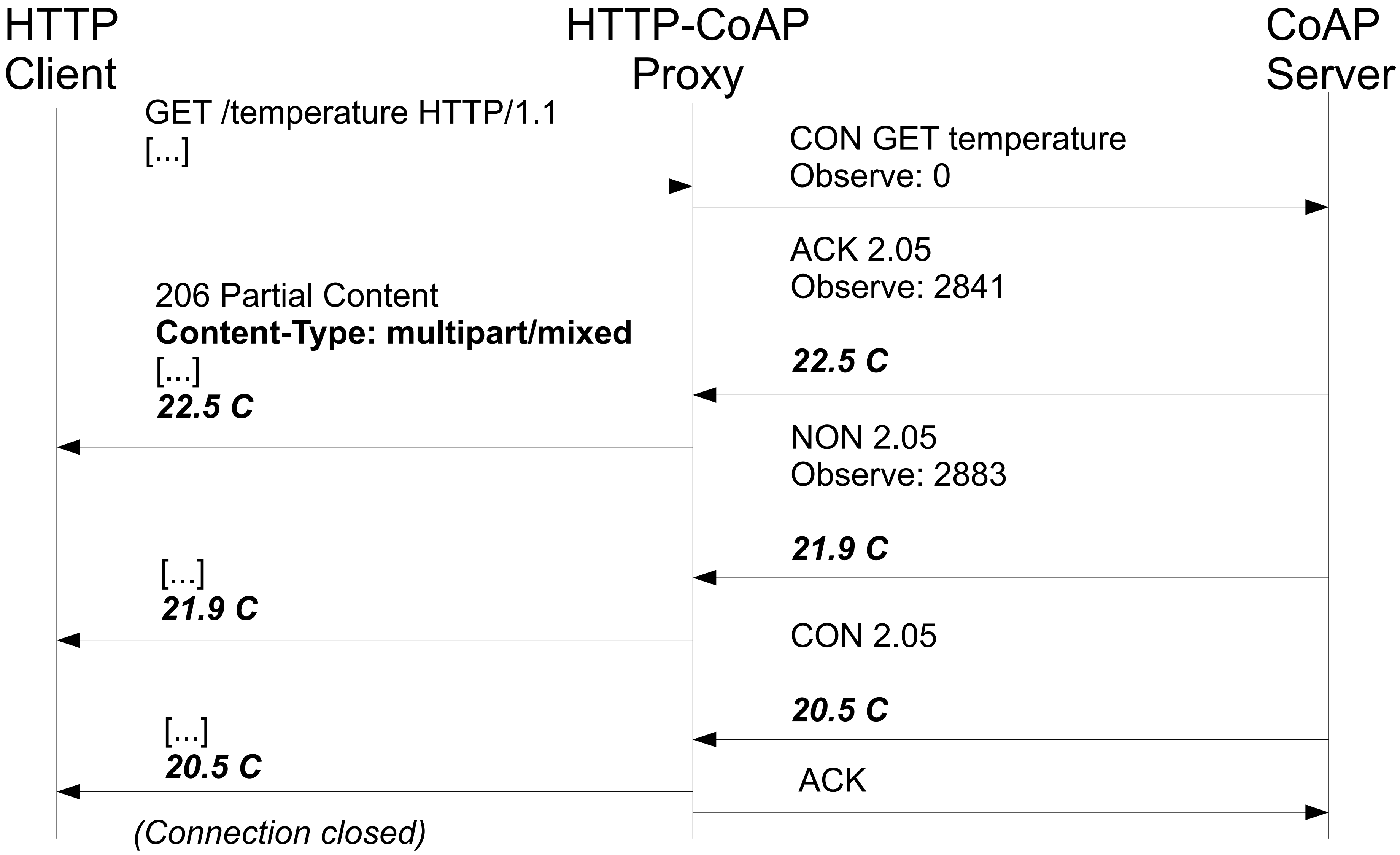
draft-castellani-core-http-mapping-03

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

Draft Quick Outline

- *Implementation of URI mapping (coap: ↔ http:)*
  - When to use it. Some simple mapping techniques proposed.
- *HTTP-CoAP proxy implementation discussion*
  - Basic proxy implementation and deployment
  - Implementing congestion control using caching and others..
  - Considerations on implementing cache refresh using Observe
  - HTTP/IPv4 – CoAP/IPv6 mapping implementation
  - HTTP Unicast – CoAP Multicast mapping implementation
    - see core-groupcomm-01
  - ***HTTP Bidirectional – CoAP Observe mapping implementation***
    - see RFC6202
- *CoAP-HTTP proxy implementation discussion*
  - Basic proxy implementarion and deployment
- *Security considerations about proxy implementation*

HTTP Bidirectional – CoAP Observe: *HTTP Streaming Example*



HTTP Client

HTTP-CoAP Proxy

CoAP Server

GET /temperature HTTP/1.1
[...]

CON GET temperature
Observe: 0

ACK 2.05
Observe: 2841

22.5 C

206 Partial Content
Content-Type: multipart/mixed
[...]
22.5 C

NON 2.05
Observe: 2883

21.9 C

[...]
21.9 C

CON 2.05

20.5 C

[...]
20.5 C

ACK

(Connection closed)

Implementation experience

- Direct experience from the draft authors
  - Squid HTTP-CoAP mapping module
    - University of Padova
    - http://telecom.dei.unipd.it/iot
      - *Both Forward and Interception operation supported*
  - HTTP-CoAP proxy based on EvCoAP
    - KoanLogic, University of Bologna and Salvatore Loreto (*as individual*)
    - https://github.com/koanlogic/webthings/tree/master/bridge/sw/lib/evcoap
- The document is open for contributions from other implementers

Next Steps

- Implementation of advanced features is ongoing
  - Will produce feedback to the document soon
- Include feedback and/or contribution that we receive from implementers
- [redacted]
  - Intended status: *Informational Best Practice*
  - Purpose: *reduce arbitrary variation in behavior of proxy implementations*

Questions to WG:

- **Who thinks this can all be worked out in deployment?**
    - **i.e., is there anything we need to standardize?**

1. **Who thinks we need "best practices" in this space?**
    a. **Who thinks we should work on this now?**
    b. **later?**
2. **Who would like to review such a spec?**
3. **Who would like to contribute to such a spec?**
4. **Who has read draft-castellani-...mapping…?**
5. **Who thinks that the draft would be a good basis for this work?**

Naïve client

draft-castellani-lwig-coap-separate-responses-00

```
C                 S
| CON MID=0x1234 |
| PUT /increment |
|--------------->| server processing
| ACK MID=0x1234 | starts.
|    X<----------|
|                |
| CON MID=0x1234 |
| PUT /increment |
|---------->X    |
|                |
| CON MID=0x1234 |
| PUT /increment |
|--------------->|
| ACK MID=0x1234 |
|    X<----------|
|                | client
|                | gives up
(continues..)
```

```
  (..continued)
| CON MID=0x1235 | client issues
| PUT /decrement | a new request
|-------->X      |
|                |
| CON MID=0xfefe | server
| 2.04 "Done"    | processing ends.
|<---------------|
| ACK MID=0xfefe |
|--------------->| inconsistency!
```

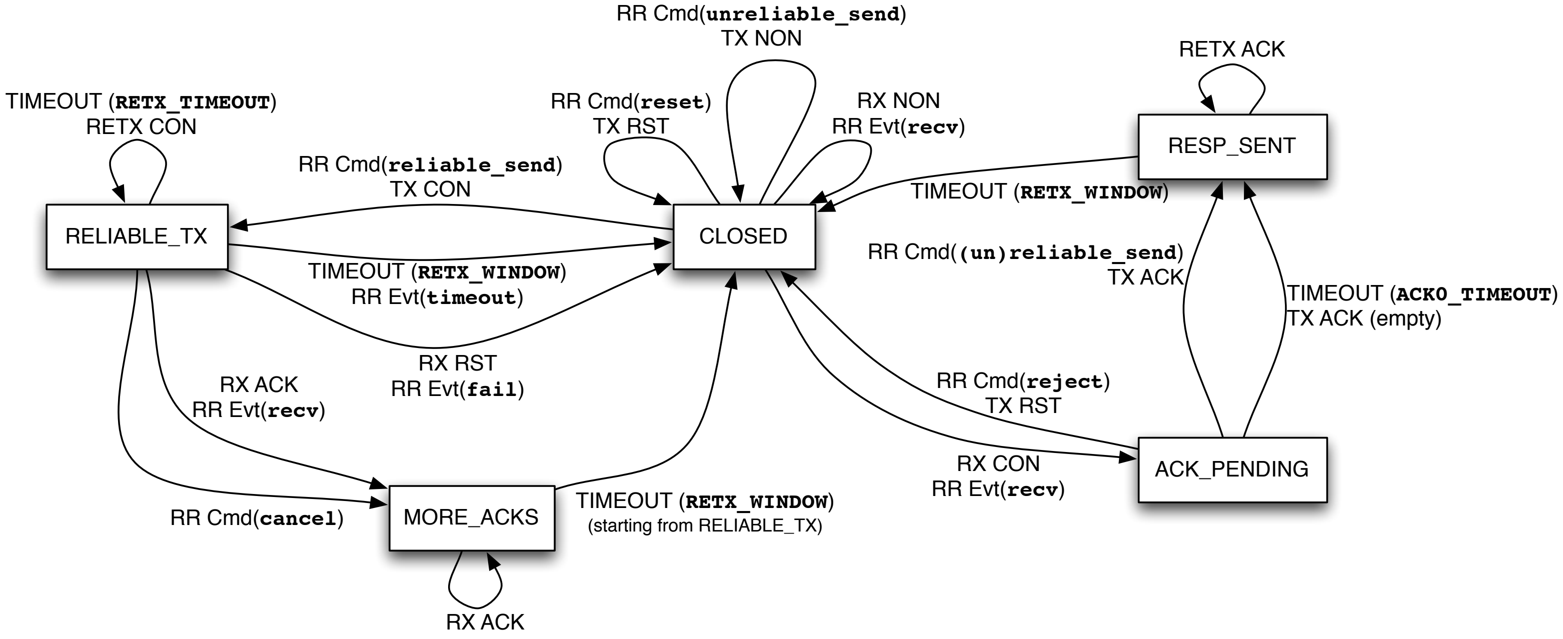
Inexperienced client

draft-castellani-lwig-coap-separate-responses-00

```
C                S
| CON MID=0x1234 |
| PUT /increment |
|---------------->|
| ACK MID=0x1234 |
|<----------------|
|                |
| CON MID=0xfefe |
| 2.04 "Done"    |
|<----------------|
| ACK MID=0xfefe |
|---------->X     |
|                |
| CON MID=0x1235 |
| PUT /decrement | client issues a new request
|---------->X     |
|                |
| CON MID=0xfefe | server retransmits the response
| 2.04 "Done"    |
|<----------------|
| ACK MID=0xfefe | client deduplication did not work
|---------------->|
```

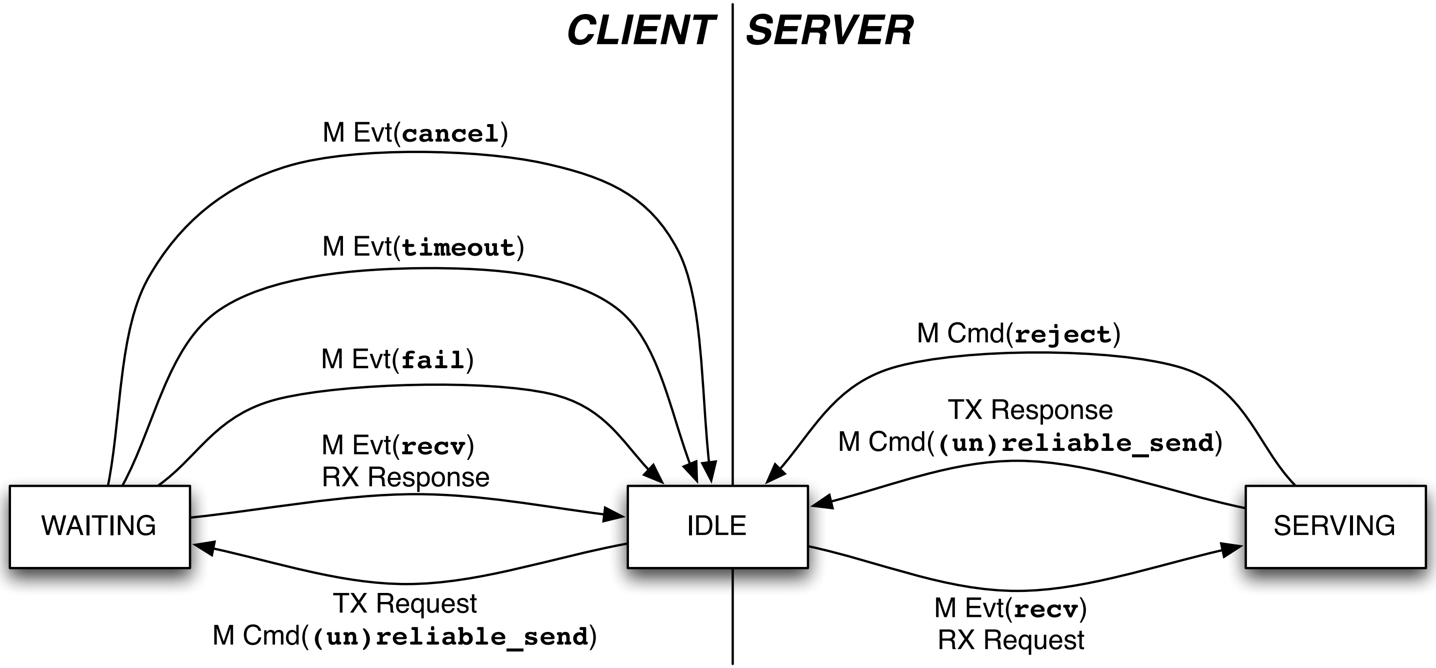
Message-layer FSM

```
RETX_TIMEOUT = RESPONSE_TIMEOUT * [ 1, RESPONSE_RANDOM_FACTOR] * 2^(RETX-1)
 RETX_WINDOW = RESPONSE_TIMEOUT * RESPONSE_RANDOM_FACTOR * 2^(MAX_RETRANSMIT-1)
ACK0_TIMEOUT = RESPONSE_TIMEOUT * ACK0_TIMEOUT_REDUCTION (<1)
```



No draft on this.

Request/response-layer FSM



No draft on this.

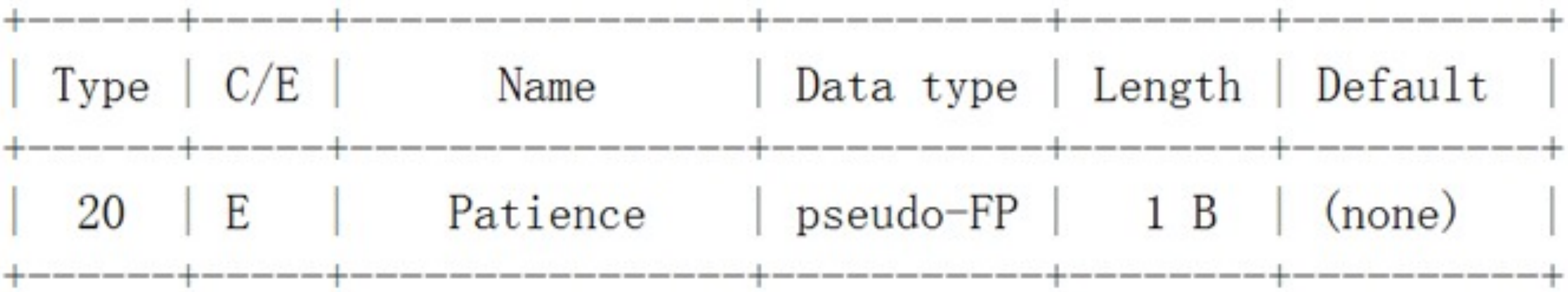
CoAP Patience Option Extension

draft-li-core-coap-patience-option-00

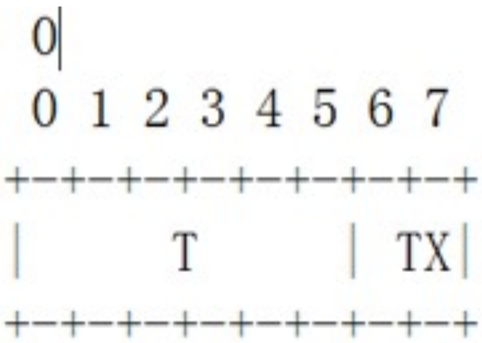
Kepeng Li
Bert Greevenbosch
Esko Dijk
Salvatore Loreto

Patience Option

✓ Definition



Type	C/E	Name	Data type	Length	Default
20	E	Patience	pseudo-FP	1 B	(none)

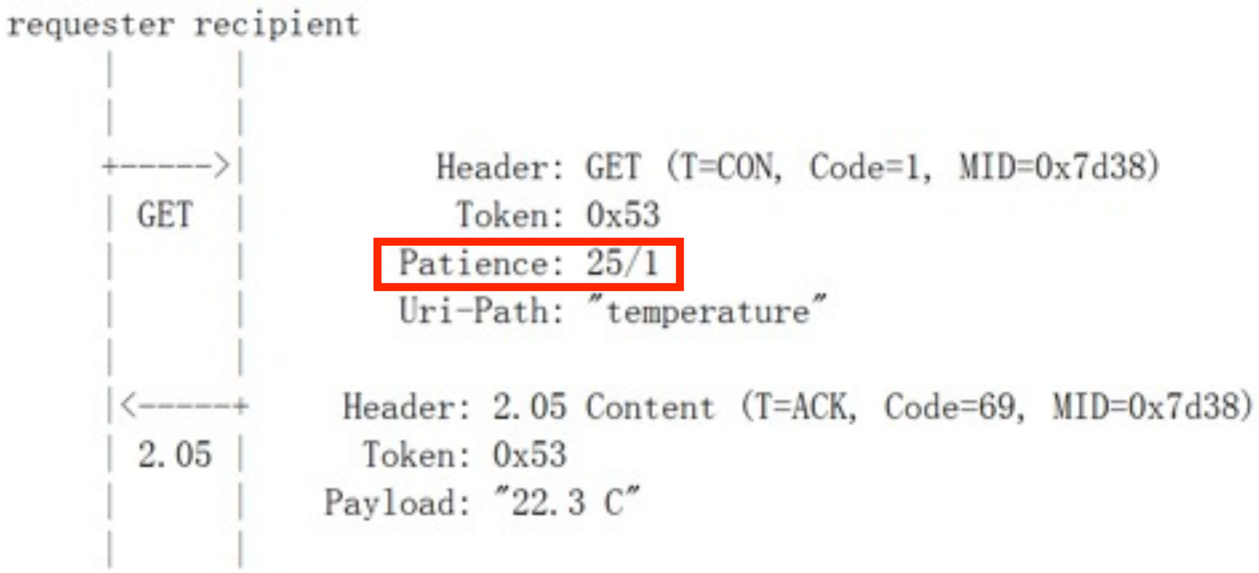


```
  0
  0 1 2 3 4 5 6 7
 +-+-+-+-+-+-+-+-+
 |      T    | TX|
 +-+-+-+-+-+-+-+-+
```

T = Time
TX = Time Exponent

Patience time = 2^(TX * 4 + 3) * T

Usage 1: in unicast request



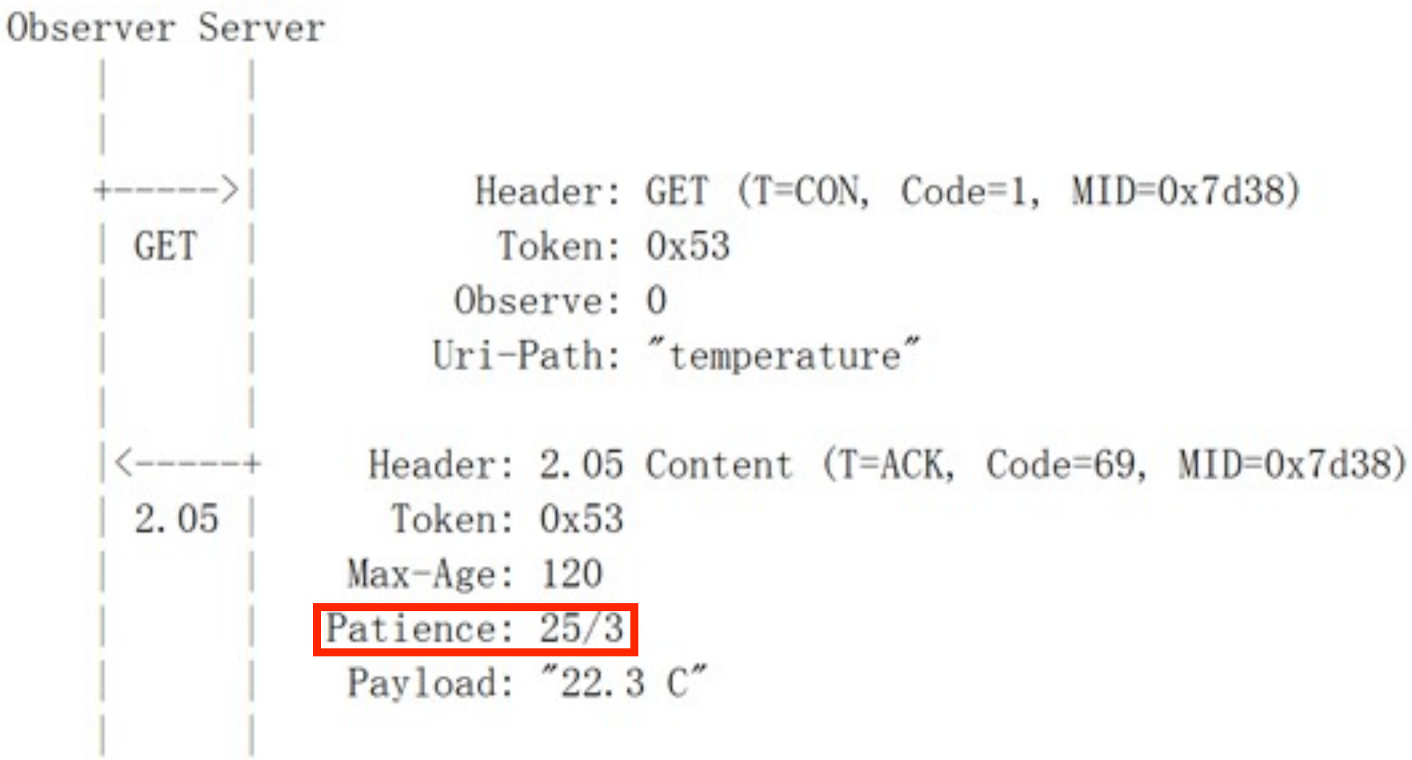
```
requester recipient
     |         |
     |         |
     +------>|          Header: GET (T=CON, Code=1, MID=0x7d38)
     | GET   |           Token: 0x53
     |         |        Patience: 25/1
     |         |        Uri-Path: "temperature"
     |         |
     |<------+      Header: 2.05 Content (T=ACK, Code=69, MID=0x7d38)
     | 2.05  |        Token: 0x53
     |         |     Payload: "22.3 C"
     |         |
```

- ✓ Usage
  - ✓Indicate the maximum time a requester is prepared to wait for a response.
- ✓Benefit
  - ✓It can avoid that the recipient wastes resources by sending a response which already exceeds the set patience timeout.

Usage 2: in multicast request

- ✓ Usage
  - ✓Used in a CoAP request to indicate that the response SHOULD be replied with a dithered delay, i.e. a randomly chosen delay between 0 and the time indicated in the option.
- ✓Bebefit
  - ✓Helps avoiding congestion i.e. multicast response storms in constrained networks.

Usage 3: observe response



```
Observer  Server
   |        |
   |        |
   +------->|          Header: GET (T=CON, Code=1, MID=0x7d38)
   | GET    |           Token: 0x53
   |        |         Observe: 0
   |        |        Uri-Path: "temperature"
   |        |
   |<-------+      Header: 2.05 Content (T=ACK, Code=69, MID=0x7d38)
   | 2.05   |       Token: 0x53
   |        |     Max-Age: 120
   |        |    Patience: 25/3
   |        |     Payload: "22.3 C"
   |        |
```

✓ Server to indicate that, after the period of time in the Max-Age option has expired, a new notification will be sent within the time interval.

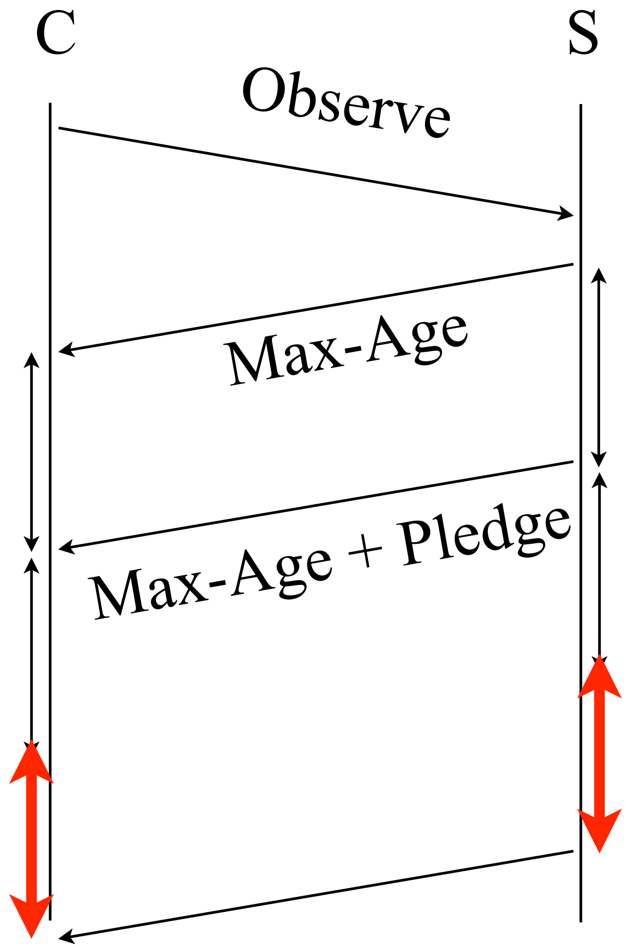
✓Benefit: maintain a robust observation relationship; avoid network congestion issues.

Open Issues

- ✓ How to differentiate multicast/unicast request?
- ✓ Do we need to use different options for different usages?
- ✓ Improve the algorithm to calculate the patience time?

Robust Observation Relationships (#174)

- **We have the 80 % solution**
- **What about the other 20 %**
  - **Pledge?**
  - **Server-side Patience?**
  - **Need to work out exact impact in caching/intermediaries**



License for waiting patiently

Questions to WG:

1. **Who thinks client-side patience options would be a useful specification?**
   1.a.**Who thinks we should work on this now?**
   1.b.**later?**
2. **Who thinks server-side patience/leisure options would be a useful specification?**
   2.a. **Who thinks we should work on this now?**
   2.b. **later?**

3. **Who would like to review such a spec?**
4. **Who would like to contribute to such a spec?**
5. **Who has read draft-li-...patience...**

Transport of CoAP over SMS, USSD and GPRS draft-becker-core-coap-sms-gprs-01

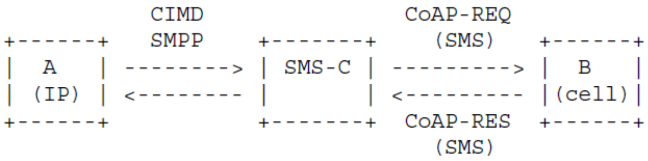
+ draft-bormann-coap-misc-13 A.4

Markus Becker, Kepeng Li,
Koojana Kuladinithi, Thomas Pötsch

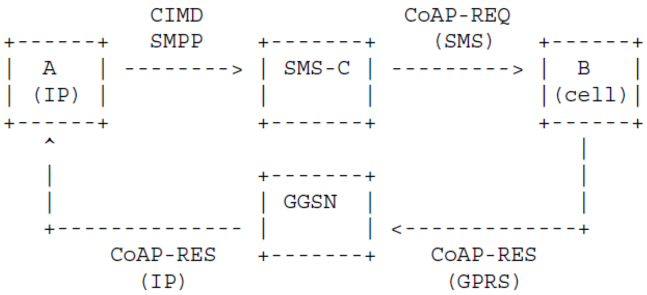
CoRE WG, IETF-83, Paris

Scenarios

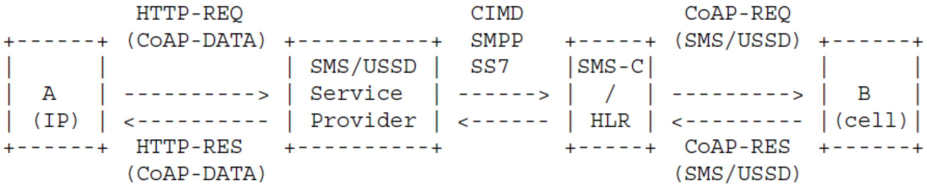
- In M2M communication, IP connectivity is not **always** supported by the constrained end-points
    - Power saving
    - Coverage (GPRS, 3G, LTE)
- SMS and USSD based communication is almost **always** supported



```
          CIMD                CoAP-REQ
+------+  SMPP     +-------+   (SMS)     +------+
|  A   | --------> | SMS-C | ----------> |  B   |
| (IP) | <-------- |       | <---------- |(cell)|
+------+           +-------+  CoAP-RES   +------+
                               (SMS)
```



```
          CIMD                CoAP-REQ
+------+  SMPP     +-------+   (SMS)     +------+
|  A   | --------> | SMS-C | ----------> |  B   |
| (IP) |           |       |             |(cell)|
+------+           +-------+             +------+
   ^                                        |
   |               +-------+                |
   |               | GGSN  |                |
   +-------------- |       | <--------------+
       CoAP-RES    +-------+    CoAP-RES
         (IP)                    (GPRS)
```



```
          HTTP-REQ                CIMD         CoAP-REQ
+------+ (CoAP-DATA) +----------+ SMPP  +-----+ (SMS/USSD) +------+
|      |             | SMS/USSD | SS7   |SMS-C|            |      |
|  A   | ----------> | Service  | ----> |  /  | ---------> |  B   |
| (IP) | <---------- | Provider | <---- | HLR | <--------- |(cell)|
+------+  HTTP-RES   +----------+       +-----+  CoAP-RES  +------+
         (CoAP-DATA)                             (SMS/USSD)
```

Technical Options

- 7, 8, 16 bit encoding of SMS
    - 7 bit: supported by all devices, 160 chars, binary data needs be encoded, e.g. Base64 (RFC4648), draft-bormann-coap-misc-13 A.4
    - 8 bit: no re-encoding necessary, 140 chars/octets, not well-supported in devices
- Larger Payload: SMS concatenation or coap-block?
    - no concatenation for USSD/GPRS -> coap-block

Technical Options

- Uri-Host and Uri-Port options SHOULD only be included when proxying. Addressing by e.g. MSISDN
- Higher default RESPONSE_TIMEOUT
- No Multicast
- New Options: Reply-To-Uri-Host/Port for GPRS return path (coap-misc Vendor-Defined Options?)
- Proxying
  - CoAP-CoAP Proxy on Mobile into LLN
  - CoAP-CoAP or CoAP-HTTP Proxy in network provider realm

Questions to WG:

1. **Who thinks CoAP over SMS would be a useful specification?**

2. **Who would like to review such a spec?**
3. **Who would like to contribute to such a spec?**

4. **Who has read draft-becker-...sms...**
5. **Who thinks that the draft would be a good basis for this work?**

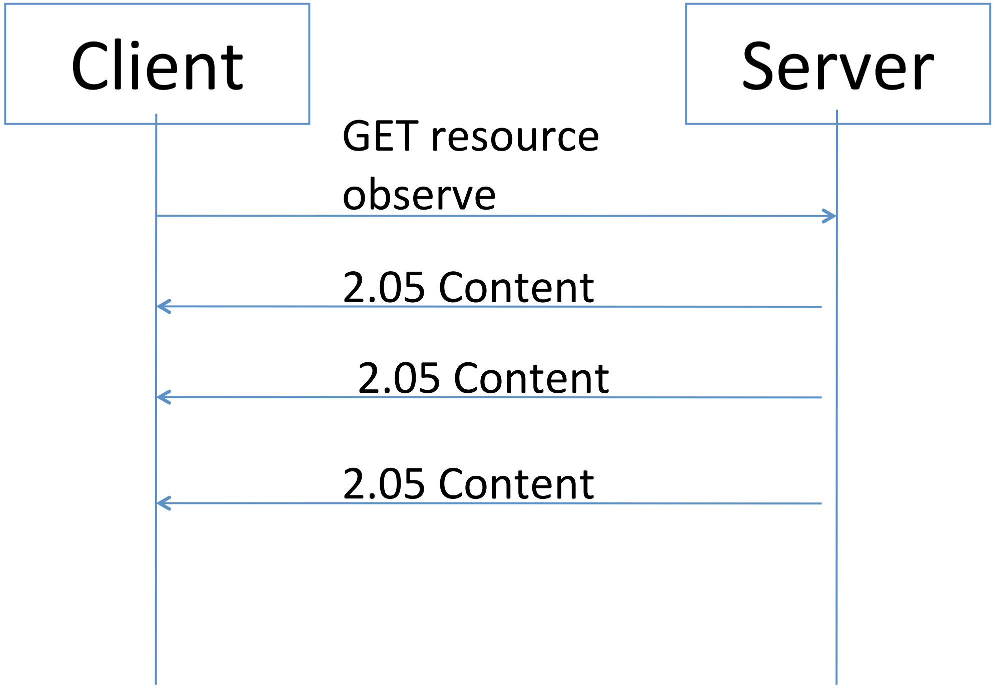
CoAP conditional observe

draft-li-core-conditional-observe-01.txt

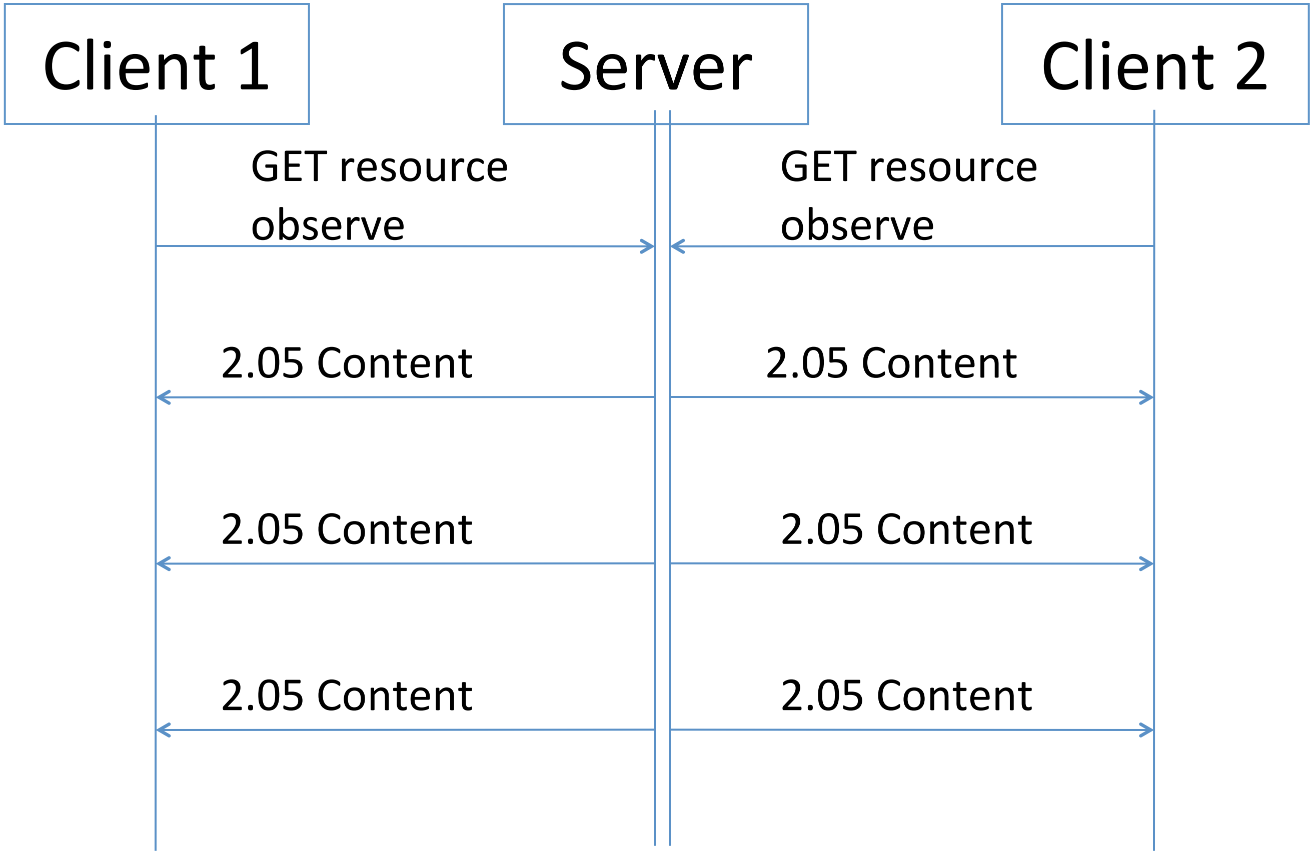
Shitao li

J.Hoebeke

Current observe mechanism :



Current Issue

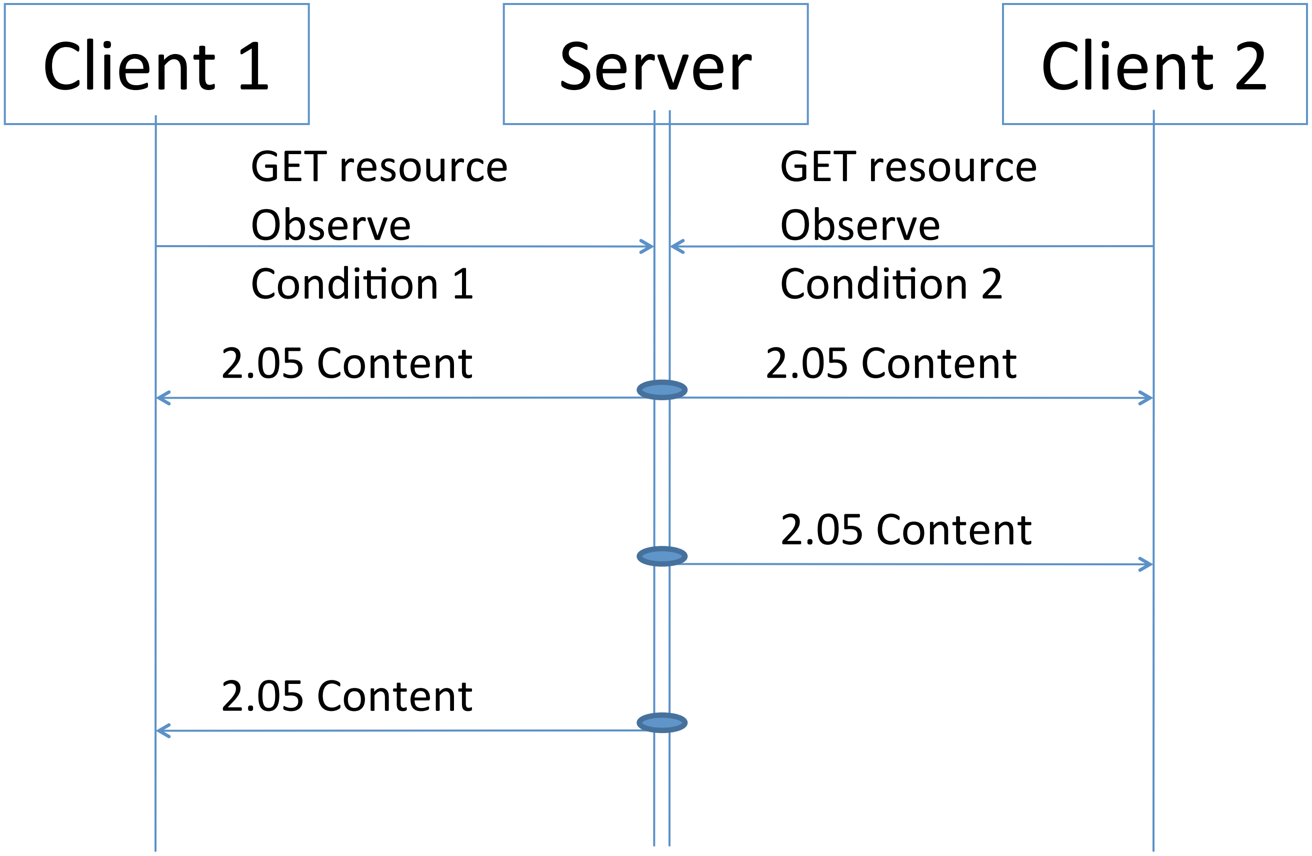


There are two clients that both observe the same resource on the same server. So the clients will receive the same response for notification. However the two clients may use the resource for different purpose, its requirements may not be the same. That is to say, not all the responses have the same meaning for both clients.

idea

- Adding condition into Observe request
- Conditions can be :
  - Minimum/Maximum Period:
    - the minimum/maximum time in seconds between notifications
  - Step:
    - how much the value of a resource should change before sending a new notification
  - Periodic:
    - periodic interval with which new notification should be sent

Condition adds into the observe

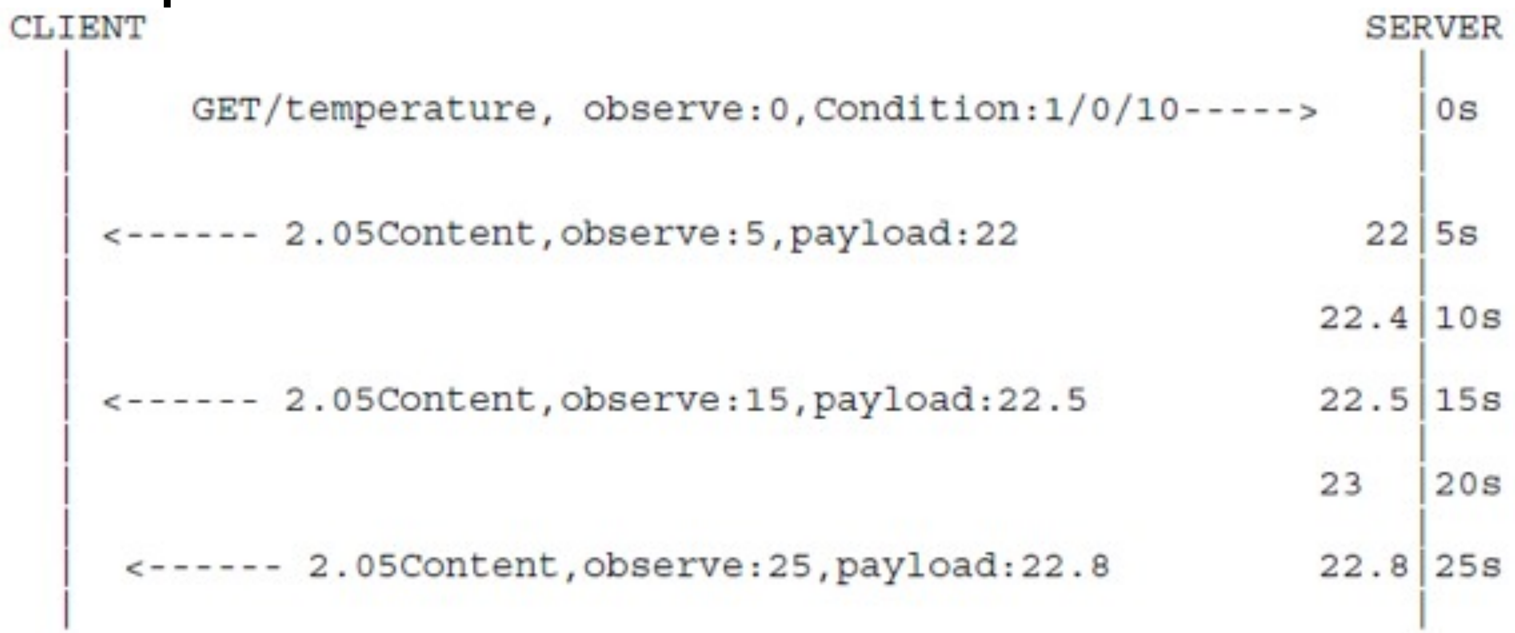


In this case, the client can send a GET request with observe and also adding the condition into the request.

when server receive such request, it will send the response to the client based on the condition received in the request, and only when the condition meets, the server will send the response.

Minimum response time

- Example 1

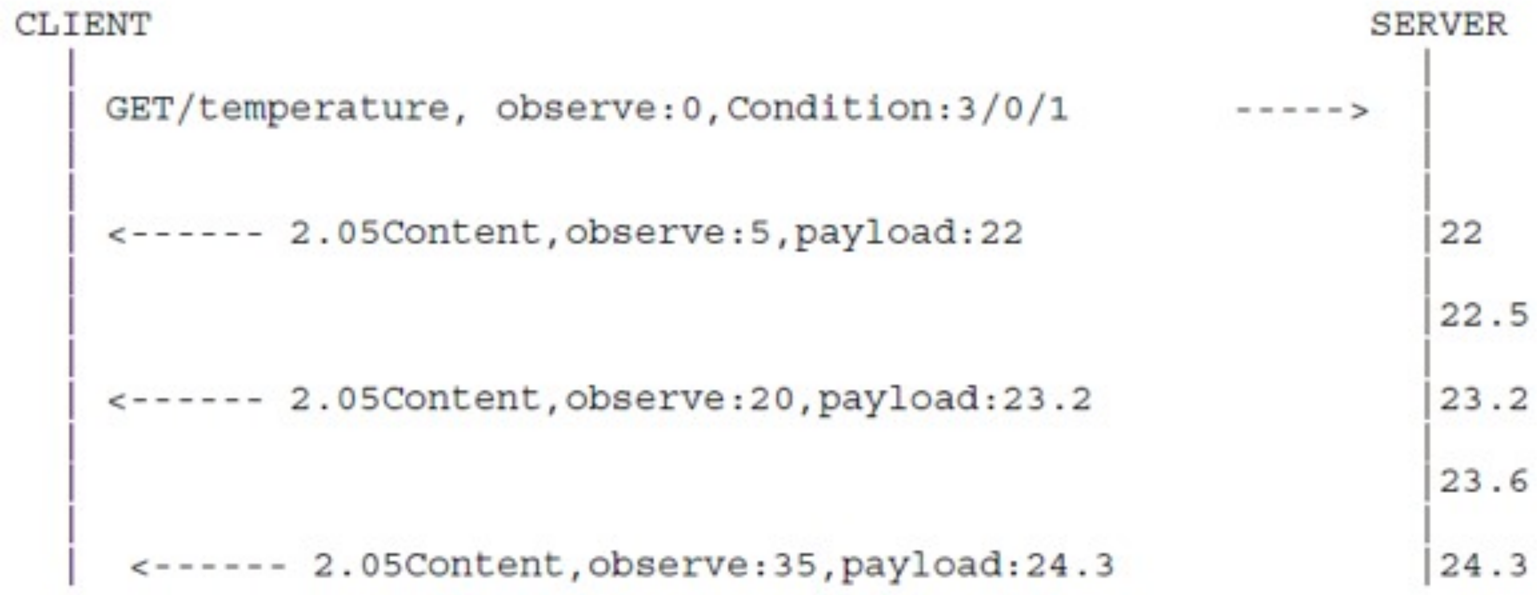


```
CLIENT                                                          SERVER
  |                                                               |
  |     GET/temperature, observe:0,Condition:1/0/10----->         |0s
  |                                                               |
  |                                                               |
  | <------ 2.05Content,observe:5,payload:22                   22 |5s
  |                                                               |
  |                                                         22.4 |10s
  |                                                               |
  | <------ 2.05Content,observe:15,payload:22.5             22.5 |15s
  |                                                               |
  |                                                         23   |20s
  |                                                               |
  |  <------ 2.05Content,observe:25,payload:22.8            22.8 |25s
  |                                                               |
```

In this example, the server collects data every 5 seconds, but the client does not want to receive the response such often, so the condition set by the client is the minimum response time , and sets to10s, then the server will send the response every 10s.

step

- Example 2



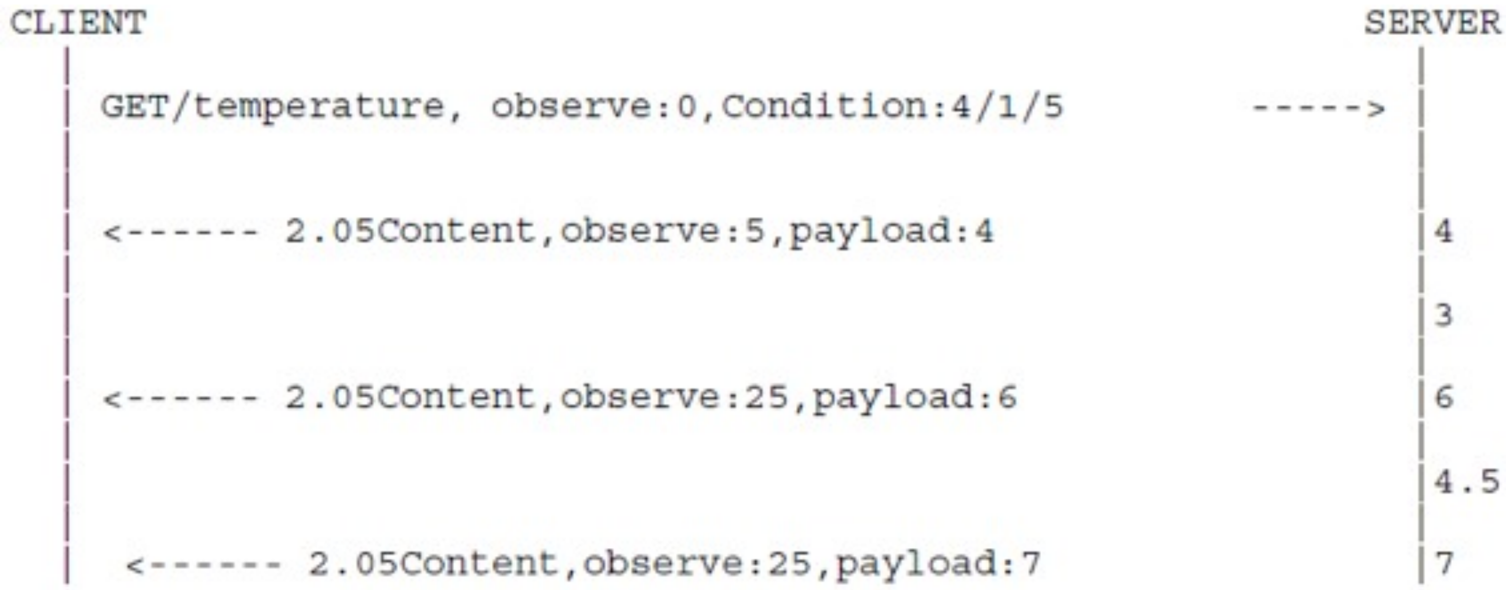
```
CLIENT                                                          SERVER
  |                                                                |
  | GET/temperature, observe:0,Condition:3/0/1            ----->   |
  |                                                                |
  |                                                                |
  | <------ 2.05Content,observe:5,payload:22                       |22
  |                                                                |
  |                                                                |22.5
  |                                                                |
  | <------ 2.05Content,observe:20,payload:23.2                    |23.2
  |                                                                |
  |                                                                |23.6
  |                                                                |
  |  <------ 2.05Content,observe:35,payload:24.3                   |24.3
```

In this example, the server works as a temperature sensor, and it collects data every 5 seconds, its precision is 0.1C.

The client does not want to receive the response such often, it does not care about temperature change smaller than 1 C either, so it set the condition accordingly.

Range

- Example 3



```
CLIENT                                                        SERVER
   |                                                             |
   | GET/temperature, observe:0,Condition:4/1/5          -----> |
   |                                                             |
   |                                                             |
   | <------ 2.05Content,observe:5,payload:4                     |4
   |                                                             |
   |                                                             |3
   |                                                             |
   | <------ 2.05Content,observe:25,payload:6                    |6
   |                                                             |
   |                                                             |4.5
   |                                                             |
   |  <------ 2.05Content,observe:25,payload:7                   |7
```

In this example, the server works as a temperature sensor, its value can be change from -10 to 50 C.

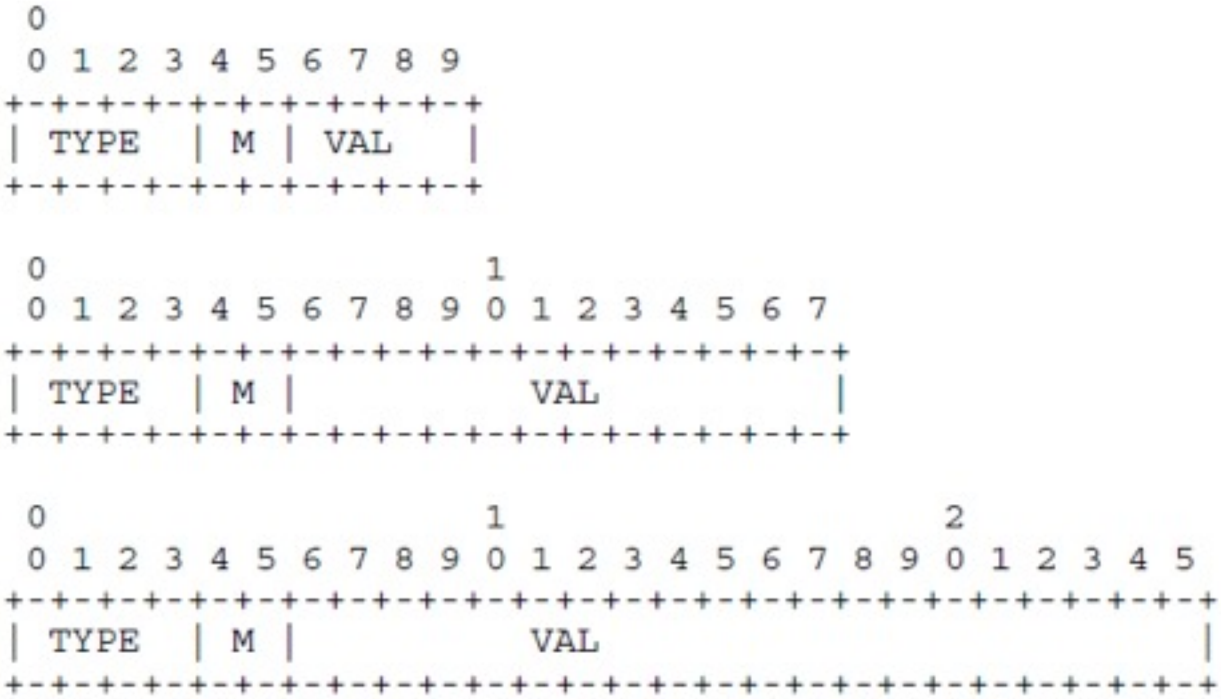
According to the application requirements, the client only wants to receive the response from the server when the temperature beyond 5 C.

How to add condition

- Adding Condition option into CoAP protocol

✓ Definition

Type	C/E	Name	Data type	Length	Default
22	E	Condition	unit	1-3 B	

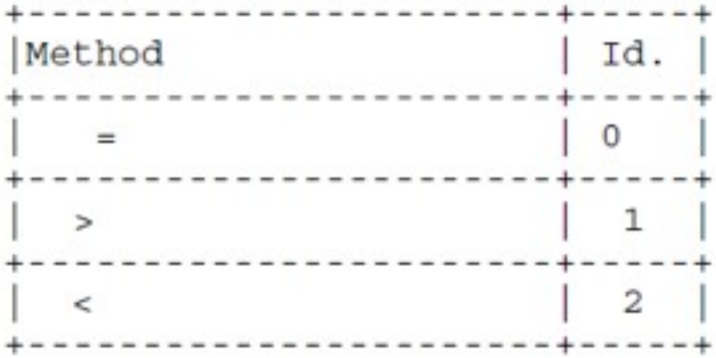


```
 0
 0 1 2 3 4 5 6 7 8 9
+-+-+-+-+-+-+-+-+-+-+
| TYPE    | M | VAL   |
+-+-+-+-+-+-+-+-+-+-+

 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| TYPE    | M |         VAL           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

 0                   1                   2
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| TYPE    | M |         VAL                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Condition type	ID
Minimum response time	1
maximum response time	2
step	3
range	**4**
periodic	**5**



Method	Id.
=	0
>	1
<	2

Value: The value can range from 0 to 2^4 (16), from 0 to 2^12 (4096) or from 0 to 2^20 (1048576).

Why not a new Condition option

- Why using a new Condition option?
  - By using a new Condition option, it can explicitly indicate the condition in observe request. What the server needs to do, is to analyze the Condition option.
  - Uri-query as described in (I-D.shelby-core-interfaces) also considered as a way to indicate the condition in Observe request. But uri-query can be used for many purposes, not only for Observe, when this is mixed with resource-specific URI-queries, this would complicate processing. . A nice split between both makes sense. Server can then do global management of all conditional observers over all resources.

Example request

Header: GET (T=CON, Code=1, MID=0x1633)

Token: 0x4a

Uri: coap://sensor.example/temperature

Observe: 0

Condition: 1/0/10

Header: GET (T=CON, Code=1, MID=0x1633)

Token: 0x4a

Uri: coap://sensor.example/temperature

Observe: 0

Condition: 3/0/1

Other suggestion

- It is also suggested that during the resource discovery procedure, the client can get the detail data information about the resource, e.g. the unit , the precision , the range, the sample time of the data, so the client can set the correct condition suit the resource.
- We can provide a value to the "obs" attribute defined in [I-D.ietf-core-observe], to indicate the conditional capabilities of a resource. In order to describe which of the 2^4 possible condition types a resource supports, a 16-bit value can be used where a bit-value of 1 at position X (from right to left) indicates that the condition type X is supported.

Questions to WG:

- **Who thinks this kind of request is best done in URI path and query components instead of options?**

1. **Who thinks observe condition options would be a useful specification?**
   a. **Who thinks we should work on this now?**
   b. **later?**
2. **Who would like to review such a spec?**
3. **Who would like to contribute to such a spec?**
4. **Who has read draft-li-...conditional...**
5. **Who thinks that the draft would be a good basis for this work?**

83rd IETF: core WG Agenda

09:00	Introduction, Agenda, Status	Chairs (10)
~~09:10~~	~~SOS Ws Report~~	~~HT (10)~~
09:20	ETSI Plugtest Report	CB (10)
09:30	1 – core Link Format	ZS (90)
09:50	1 – core, block, observe, WGLC	ZS, KH, CB (20)
10:45	Interfaces	ZS (45)
11:30	retire to Friday, 09:00 Intro	Chairs (05)
09:05	SOS Ws Report for real	HT (10)
09:15	1 – core, block, observe, WGLC	(15)
09:30	Group 4 – DNA	MI PV (30)
10:00	Group 5 – “sleepy”	MV TF (30)
10:30	Group 6 – “other”	AC KL MB SL (20)
10:50	Group 7 – “next steps”	Chairs (10)
11:00	retire	

Group 7: recharter?

CoRE WG

- Getting closer to finishing its charter
- What *needs* to be done to complete this picture?

- 1) Declare success and close the WG
- 2) Focus on a specific open problem
  - 2a) and do some maintenance for a while
- 0) Boil the ocean

Areas of work (not all *in* CoRE)

1. **Smart Object Lifecycle Architecture (SOLAr)**
   - **(i.e., security work without inventing security mechanisms)**
2. **CoAP over X (e.g., SMS, TCP, XMPP…)**
3. **CoAP in wider RESTful architectures**
   - **e.g. working with JSON, HTTP, HTTP 2.0**
4. **CoAP for wider communication models (“sleepy”)**
5. **Discovery/Interfaces/Intermediaries, Profiles, ...**
6. **Options, more options, give me more options!**

83rd IETF: core WG Agenda

09:00	Introduction, Agenda, Status	Chairs (10)
~~09:10~~	~~SOS Ws Report~~	~~HT (10)~~
09:20	ETSI Plugtest Report	CB (10)
09:30	1 – core Link Format	ZS (90)
09:50	1 – core, block, observe, WGLC	ZS, KH, CB (20)
10:45	Interfaces	ZS (45)
11:30 retire to Friday, 09:00	Intro	Chairs (05)
09:05	SOS Ws Report for real	HT (10)
09:15	1 – core, block, observe, WGLC	(15)
09:30	Group 4 – DNA	MI PV (30)
10:00	Group 5 – “sleepy”	MV TF (30)
10:30	Group 6 – “other”	AC KL MB SL (20)
10:50	Group 7 – “next steps”	Chairs (10)

11:00 retire