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**

Drafts

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

Working Group Documents:

Draft name	Rev.	Dated	Status	Comments, Issues
Active:				
draft-ietf-core-link-format	-08	new 2011-11-14	Active	0/23
draft-ietf-core-observe	-03	2011-10-31	Active	0/19
draft-ietf-core-coap	-08	2011-10-31	Active	3/92
draft-ietf-core-block	-04	2011-07-11	Active	3/19

Related Active Documents (not working group documents):

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

draft-vanderstok-core-bc	-05	2011-10-31
draft-shelby-core-resource-directory	-02	2011-10-31
draft-sarikaya-core-sbootstrapping	-03	2011-10-31
draft-nieminen-core-service-discovery	-01	2011-10-31
draft-garcia-core-security	-03	2011-10-31
draft-castellani-core-http-mapping	-02	2011-10-31
draft-bormann-coap-misc	-09	2011-10-31
draft-arkko-core-dev-urn	-01	2011-10-31
draft-ma-core-dhcp-pd	-00	2011-10-24
draft-he-core-energy-aware-pd	-00	2011-10-24
draft-cao-core-pd	-00	2011-10-24
draft-becker-core-coap-sms-gprs	-00	2011-10-24
draft-li-core-coap-size-option	-02	2011-10-21
draft-li-core-coap-over-sms	-00	2011-10-21
draft-yegin-coap-security-options	-00	2011-10-17
draft-rahman-core-groupcomm	-07	2011-10-12

+ draft-jennings-senml-07.txt

Thu
Fri
Not discussed

Status link-format

- -02 Completed WGLC on Jan 26
- WGLC comments addressed in -03 (March 14)
- Editorial fixes, examples, post-Prague (-04)
- Examples updated for new -core response codes (-05)
- Clarify UTF-8 questions (-06)
- Content moved around between link-format, -core, -observe (-07)
- Did not catch some ID-nits, being fixed (-08, -09)

➼ One-week last-call this week

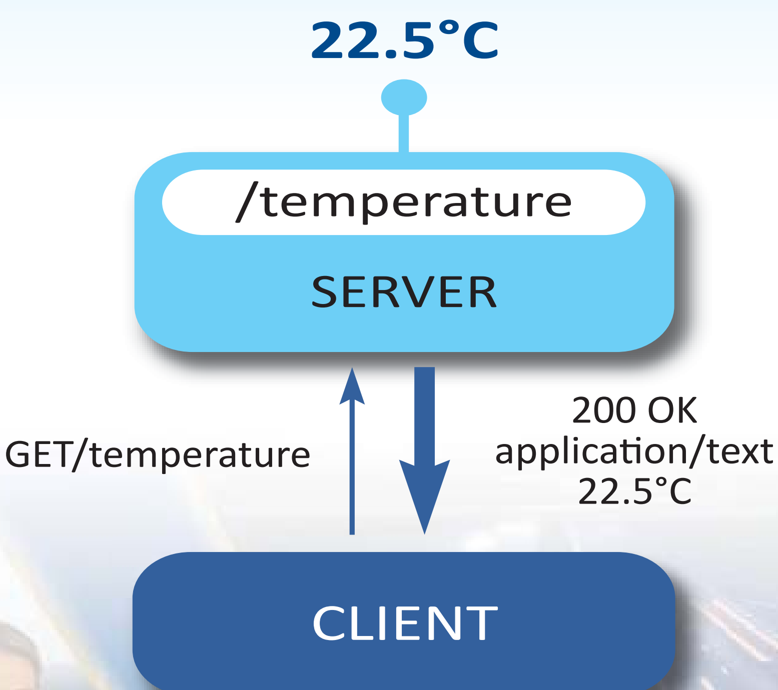
➼ Barring surprises, submit to IESG on Nov 28

IoT CoAP Plugtests

24-25 March 2012, Paris

Registration Deadline: 9 March 2012

Website: http://www.etsi.org/plugtests



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.

random implementation report: tinyDTLS

- client and server, ~2500 LOC
- DTLS v1.1 (“v1.2 ready”)
  - Have: TLS_PSK_WITH_AES_128_CBC_SHA
  - Working on: TLS_PSK_WITH_AES_128_CCM_8
- Contiki support being worked on
- Work in progress (at very early stage)
  - record layer and handshake only
  - Proof of concept, not (yet) ready for prime time!
- Code & docs at `tinydtls.sourceforge.net`

82nd IETF: core WG Agenda

15:20	Introduction, Agenda, Status	Chairs (10)
15:30	1 – core CoAP	ZS (90)
17:00	1 – block, observe	CB (20)
17:20 retire to Friday, 11:20	Intro	Chairs (05)
11:25	1 – core CoAP: artificial limitations	CB (10)
11:35	Group Communication	AR (10)
11:43	Naming, Data Formats	JA (10)
11:57	Discovery	ZC MI ZS
12:27	Using CoAP/Naming	KL (8)
12:35	HTTP Mapping	AR (5)
12:40	security	SLK AY BS
13:10	new: CoAP-over-X	KL (10)
13:20	next steps	Chairs (10)
13:30 retire		

82^{nd} IETF: core WG Agenda

15:20	Introduction, Agenda, Status	Chairs (10)
15:30	1 – core CoAP	ZS (90)
17:00	1 – block, observe	CB (20)
17:20 retire to Friday, 11:20	Intro	Chairs (05)
11:25	1 – core CoAP: artificial limitations	CB (10)
11:35	Group Communication	AR (10)
11:43	Naming, Data Formats	JA (10)
11:57	Discovery	ZC MI ZS
12:27	Using CoAP/Naming	KL (8)
12:35	HTTP Mapping	AR (5)
12:40	security	SLK AY BS
13:10	new: CoAP-over-X	KL (10)
13:20	next steps	Chairs (10)
13:30 retire		

Constrained Application Protocol

draft-ietf-core-coap-08

Z. Shelby, K. Hartke, C. Bormann, B. Frank

CoRE WG, IETF-82 Taipei

Progress Since Quebec

- One revision of the draft (coap-08)
- Closed 5 tickets (minor changes)
- Use by other SDOs
  - ETSI M2M Release 1 approved
    - Includes a CoAP binding
- Recent CoAP tools
  - 2 Contiki CoAP implementations
  - JCoAP project
  - TinyDTLS project

Tickets Closed

- Technical
  - Re-focused the security section on raw public keys (#166)
  - Added an 4.06 error to Accept (#165)
- Editorial
  - Clarified matching rules for messages (#175)
  - Fixed a bug in Section 8.2.2 on Etags (#168)
  - Added an IP address spoofing threat analysis contribution (#167)

Finishing Up

- Define response suppression for multicast
  - Some text needs to be added (See #177)
- Wrap up raw public keys
- Relax the artificial limitations

Removing Artificial Limits

- The 15 option limit has become a problem
  - See Ticket #176 for background
  - e.g. ETSI M2M running out of options
  - Splitting URI path and query into segments the main cause of the issue (but necessary for normalization)
- Option length limited to 270 bytes
  - Not clear if this ever will be a problem
  - Options can be concatenated as with Proxy-Uri

Option Limit Solution

- Keep it as now for up to 14 options
- Value OC=0b1111 indicates > 14 options
  - End of options marker used instead (0b11110000)

```
 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
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |Ver| T |  15   |      Code     |          Message ID           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |   Options ...
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |1 1 1 1 0 0 0 0|   Payload (if any) ...
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- This limits the option delta to 14
  - No problem with fence posting mechanism
- Other options discussed in draft-bormann-coap-misc-10

Wrap up Raw Public Keys

- This is what the WG decided in Quebec
- Combined the symmetric key modes
  - Now a single DTLS PreSharedKey mode
  - Key management out of scope
- RawPublicKey mode added
  - Node has an asymmetric key pair
  - TLS almost supports raw public keys
    - coap-08 defines provisional X.509 wrapping
    - But TLS WG is defining a permanent solution
      - draft-wouters-tls-oob-pubkey-01
- Appendix D defines Provisioning and ACLs
  - Hashed identity of public key (needs definition)
  - Defines use of those identities for access control

Wrap up Raw Public Keys

- Down-ref issue with TLS raw public keys
  - New draft not yet a TLS WG item
    - We are needed in the TLS WG session after this ☺
- Possible way forward
  - Spin-out a RawPublicKey WG document
    - PreSharedKey would be “must implement” in CoAP
    - Include text that new modes are expected in the future
  - Provisioning and access control
    - Hashed identity needs definition (length, hash, format)
    - Provisioning and ACLs needs more work
    - Spin this out to the RawPublicKey draft

Quality of Implementation

- It is possible to write bad implementations
  – e.g. one that never piggy-backs responses
- DO NOT DO THAT
  – unless there is a good reason
- The protocol can’t know whether that is so
  – RFC 2119 “SHOULD” would be wrong (!)

- Add a quality of implementation note?

Content-Type Registry (1)

- Today: Numeric CoAP Content-Type
  - registry maps to MIME Media-Type
  - HTTP proxy needs to know registry contents
- Issue: Content-Encoding
  - default: identity, but might want deflate, exi, ...
  - Separate CoAP option? YAGNI.
  - Better: encode this in Content-Type Option

Content-Type Registry (2)

- Rename “media-type registry” to “content-type registry”
- Add a column “content-coding”
  - usual value: “identity”
- Suggested shortcut representation:
  - application/atom+xml**@gzip**

- Do we have the right policy:
  - 0–200 Expert Review
  - 256–65535 First-come-first-served
    - Will we see vanity entries? [Siemens-Temperature]

82nd IETF: core WG Agenda

15:20	Introduction, Agenda, Status	Chairs (10)
15:30	1 – core CoAP	ZS (90)
17:00	1 – block, observe	CB (20)
17:20 retire to Friday, 11:20	Intro	Chairs (05)
11:25	1 – core CoAP: artificial limitations	CB (10)
11:35	Group Communication	AR (10)
11:43	Naming, Data Formats	JA (10)
11:57	Discovery	ZC MI ZS
12:27	Using CoAP/Naming	KL (8)
12:35	HTTP Mapping	AR (5)
12:40	security	SLK AY BS
13:10	new: CoAP-over-X	KL (10)
13:20	next steps	Chairs (10)
13:30 retire		

The observe option

- Enables the observation of a resource
  - Changes to the resource value are asynchronously reported
- Technically stable since -02 (March 15, 2011)
  - new: Max-OFE (later)

- Grand rewrite has improved readability significantly
- draft-ietf-core-observe-03.txt ready for WGLC

New: Max-OFE

- **Enables a proxy (or client-local cache) to operate on stale data optimistically**
  - **Maximum Optimistic Freshness Extension**
- **Shares motivation with RFC 5861 (for HTTP)**
  - **but reduced to the essence**
- **Split between max-age and max-ofe marks the time when a new version could meaningfully be requested**
- **Max-ofe bounds the time for a new version of the resource representation ("keepalive" for observation relationship)**

```
     Observed   CLIENT  SERVER     Actual
 t   State         |      |         State
     ____________  |      |  ____________
 1                 |      |
 2    unknown      |      |     18.5 C
 3                 +----->|                  Header: GET 0x43011633
 4                 | GET  |                   Token: 0x4a
 5                 |      |                Uri-Path: temperature
 6                 |      |                 Observe: 0
 7                 |      |
 8                 |      |
 9   ____________  |<-----+                  Header: 2.05 0x64451633
10                 | 2.05 |                   Token: 0x4a
11    18.5 C       |      |                 Observe: 9
12                 |      |                 Max-Age: 15
13                 |      |                 Max-OFE: 60
14                 |      |                 Payload: "18.5 C"
15                 |      |
16                 |      |  ____________
17   ____________  |<-----+                  Header: 2.05 0x54457b50
18                 | 2.05 |     19.2 C        Token: 0x4a
19    19.2 C       |      |                 Observe: 17
20                 |      |                 Max-Age: 15
21                 |      |                 Max-OFE: 60
22                 |      |                 Payload: "19.2 C"
23                 |      |
```

Figure 3: A client registers and receives a notification of the current state and upon a state change

```
     Observed   CLIENT  SERVER     Actual
 t   State         |       |         State
     ____________  |       |  ____________
24                 |       |
25    19.2 C       |       |      19.2 C
26                 |       |
27                 |       |
28                 |       |  ____________
29                 | X----+                   Header: 2.05 0x54457b51
30                 | 2.05 |       19.7 C        Token: 0x4a
31                 |       |                 Observe: 29
32   ____________  |       |                 Max-Age: 15
33                 |       |                 Max-OFE: 60
34    19.2 C       |       |                 Payload: "19.7 C"
35   (optimistic)  |       |
36                 |       |  ____________
37   ____________  |<-----+                   Header: 2.05 0x55457b52
38                 | 2.05 |       18.9 C        Token: 0x4a
39    18.9 C       |       |                 Observe: 37
40                 |       |                    ETag: 0x78797a7a79
41                 |       |                 Max-Age: 15
42                 |       |                 Max-OFE: 60
43                 |       |                 Payload: "18.9 C"
44                 |       |
```

Figure 4: The client optimistically assumes that the state did not change after Max-Age ended

```
    Observed   CLIENT  SERVER     Actual
 t  State        |       |         State
    ------------ |       |  ------------
45               |       |
46   18.9 C      |       |       18.9 C
47               |       |
48               |       |  ------------
49               |     CRASH
50               |
51               |
52  ------------ |
53               |
54   18.9 C      |
55  (optimistic) |
56               |
 :               :
111              |
112 ------------ |
113              |
114  18.9 C      |
115 (invalid)    |
116              |
```

Figure 5: The server crashes and leaves the client with stale information

```
CLIENT  PROXY  SERVER
   |      |      |
   +----->|      |     Header: GET 0x43011635
   | GET  |      |      Token: 0x6a
   |      |      |  Proxy-Uri: coap://sensor.example/status
   |      |      |    Observe: 0
   |      |      |
   |<- - -+      |     Header: 0x60001635
   |      |      |
   |      +----->|     Header: GET 0x4401af90
   |      | GET  |      Token: 0xaa
   |      |      |   Uri-Host: sensor.example
   |      |      |   Uri-Path: status
   |      |      |    Observe: 0
   |      |      |
   |      |<-----+     Header: 2.05 0x6445af90
   |      | 2.05 |      Token: 0xaa
   |      |      |    Observe: 67
   |      |      |    Max-Age: 60
   |      |      |    Max-OFE: 60
   |      |      |    Payload: "ready"
   |      |      |
   |<-----+      |     Header: 2.05 0x4445af94
   | 2.05 |      |      Token: 0x6a
   |      |      |    Observe: 17346
```

```
|      |      |   Observe: 17346
|      |      |   Max-Age: 60
|      |      |   Max-OFE: 60
|      |      |   Payload: "ready"
|      |      |
+- - ->|      |    Header: 0x6000af94
|      |      |
|      |<-----+    Header: 2.05 0x54455a20
|      | 2.05 |     Token: 0xaa
|      |      |   Observe: 157
|      |      |   Max-Age: 60
|      |      |   Max-OFE: 60
|      |      |   Payload: "busy"
|      |      |
|<-----+      |    Header: 2.05 0x5445af9b
| 2.05 |      |     Token: 0x6a
|      |      |   Observe: 17436
|      |      |   Max-Age: 60
|      |      |   Max-OFE: 60
|      |      |   Payload: "busy"
|      |      |
```

Figure 8: A client observes a resource through a proxy

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
- $16 \leq$ Block size ≤ 2048

Status of core-block-04

- Open issue: integrated or separate size option (later)
- No technical changes since the split Block1/Block2

- Editorial rewrite pending → –05
- Should then become ready for WGLC

CoAP Size Option Extension

draft-li-core-coap-size-option-02

Kepeng Li
Linyi Tian
Barry Leiba

Size Option

✓ Definition

Type	C/E	Name	Data type	Length	Default
12	E	Size	uint	1-4 B	

Size Option in a Post/Put request

```
CLIENT                                                    SERVER
  |                                                          |
  |                                                          |
  | CON [MID=1234], Put, /options, 1/0/128, Size: 5000 ----> |
  |                                                          |
  |  <------ ACK [MID=1234], 4.13 Request Entity Too Large   |
  |                                                          |
  |                                                          |
  |                                                          |
  |                                                          |
```

✓ Usage

- In Put/Post request:

  --Indicate the resource size

Size option in GET

✓GET request without Size option

- If size>PDU, Size option SHOULD be included in the response.
- If size<PDU, Size option MAY be included in the response.

✓GET request with Size option, without Block option:

- Size=0, only returns size info, no payload;
- Size is empty, returns size info:
  - -- if size<PDU, returns the whole payload
  - --if size>PDU, returns first Block
- Size has a value other than 0, not specified in the draft .

Integration with Block option

✓ GET request with Size option, with Block option:

- Size=0, only returns size info, no payload; this case is not specified in the draft.
- Size is empty, returns size info:

  -- if size<PDU, returns the whole payload

  --if size>PDU, returns first Block
- Size has a value other than 0, not specified in the draft .

✓ POST/PUT request with Size option, with Block option:

- indicate the size of the resource
- SHOULD be included in the first Block request

Summary

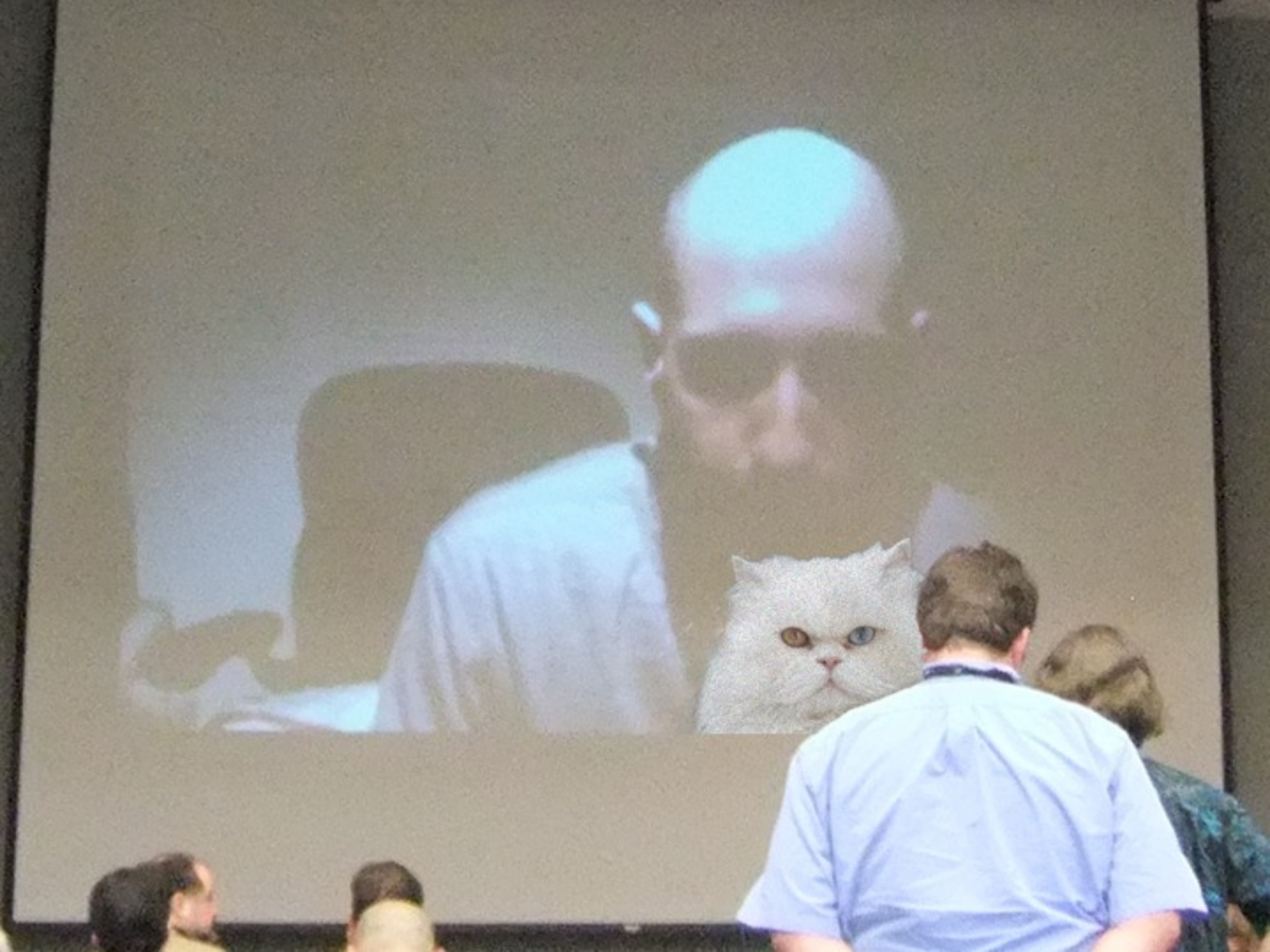
- ✓ Changes in -02
  - ✓Keep it as Elective option
  - ✓Clarified how to integrate with Block draft
  - ✓Provide the end-points with hints on when to send this and when not
  - ✓Clarified “HEAD” function and size indication
- ✓Implementation Experience
  - ✓More useful for the large resources
  - ✓More useful together with Block option
- ✓Recommendation for the next step
  - ✓Merge it with Block draft or
  - ✓A separate WG draft?

82nd IETF: core WG Agenda

15:20	Introduction, Agenda, Status	Chairs (10)
15:30	1 – core CoAP	ZS (90)
17:00	1 – block, observe	CB (20)
17:20 retire to Friday, 11:20	Intro	Chairs (05)
11:25	1 – core CoAP: artificial limitations	CB (10)
11:35	Group Communication	AR (10)
11:43	Naming, Data Formats	JA (10)
11:57	Discovery	ZC MI ZS
12:27	Using CoAP/Naming	KL (8)
12:35	HTTP Mapping	AR (5)
12:40	security	SLK AY BS
13:10	new: CoAP-over-X	KL (10)
13:20	next steps	Chairs (10)
13:30 retire		

TLS WG & Raw Keys

- The TLS WG meeting discussed the topic of doing work on raw key mode



- TLS WG discussion was strongly positive

82^{nd} IETF: core WG Agenda

15:20	Introduction, Agenda, Status	Chairs (10)
15:30	1 – core CoAP	ZS (90)
17:00	1 – block, observe	CB (20)
17:20 retire to Friday, 11:20	Intro	Chairs (05)
11:25	1 – core CoAP: artificial limitations	CB (10)
11:35	Group Communication	AR (10)
11:43	Naming, Data Formats	JA (10)
11:57	Discovery	ZC MI ZS
12:27	Using CoAP/Naming	KL (8)
12:35	HTTP Mapping	AR (5)
12:40	security	SLK AY BS
13:10	new: CoAP-over-X	KL (10)
13:20	next steps	Chairs (10)
13:30 retire		

82^{nd} IETF: core WG Agenda

15:20	Introduction, Agenda, Status	Chairs (10)
15:30	1 – core CoAP	ZS (90)
17:00	1 – block, observe	CB (20)
17:20 retire to Friday, 11:20	Intro	Chairs (05)
11:25	1 – core CoAP: artificial limitations	CB (10)
11:35	Group Communication	AR (10)
11:43	Naming, Data Formats	JA (10)
11:57	Discovery	ZC MI ZS
12:27	Using CoAP/Naming	KL (8)
12:35	HTTP Mapping	AR (5)
12:40	security	SLK AY BS
13:10	new: CoAP-over-X	KL (10)
13:20	next steps	Chairs (10)
13:30 retire		

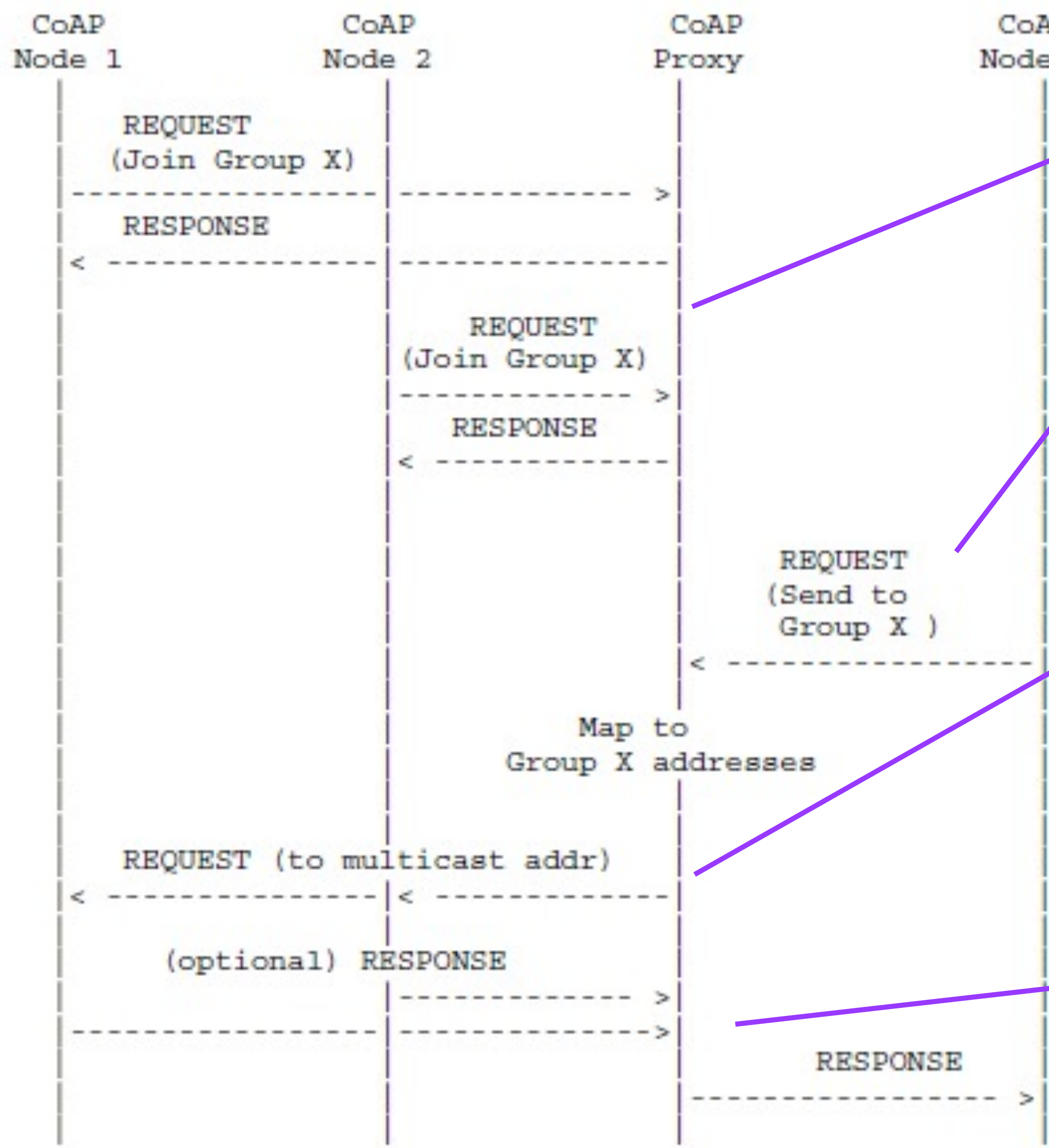
Group Communication for CoAP

Akbar Rahman
Esko Dijk

IETF 82, November 2011

http://tools.ietf.org/html/draft-rahman-core-groupcomm-07

CoAP Group Communications Concept



1) Multiple receiver nodes form a group

2) Source (sender) sends a single message with content to the group address

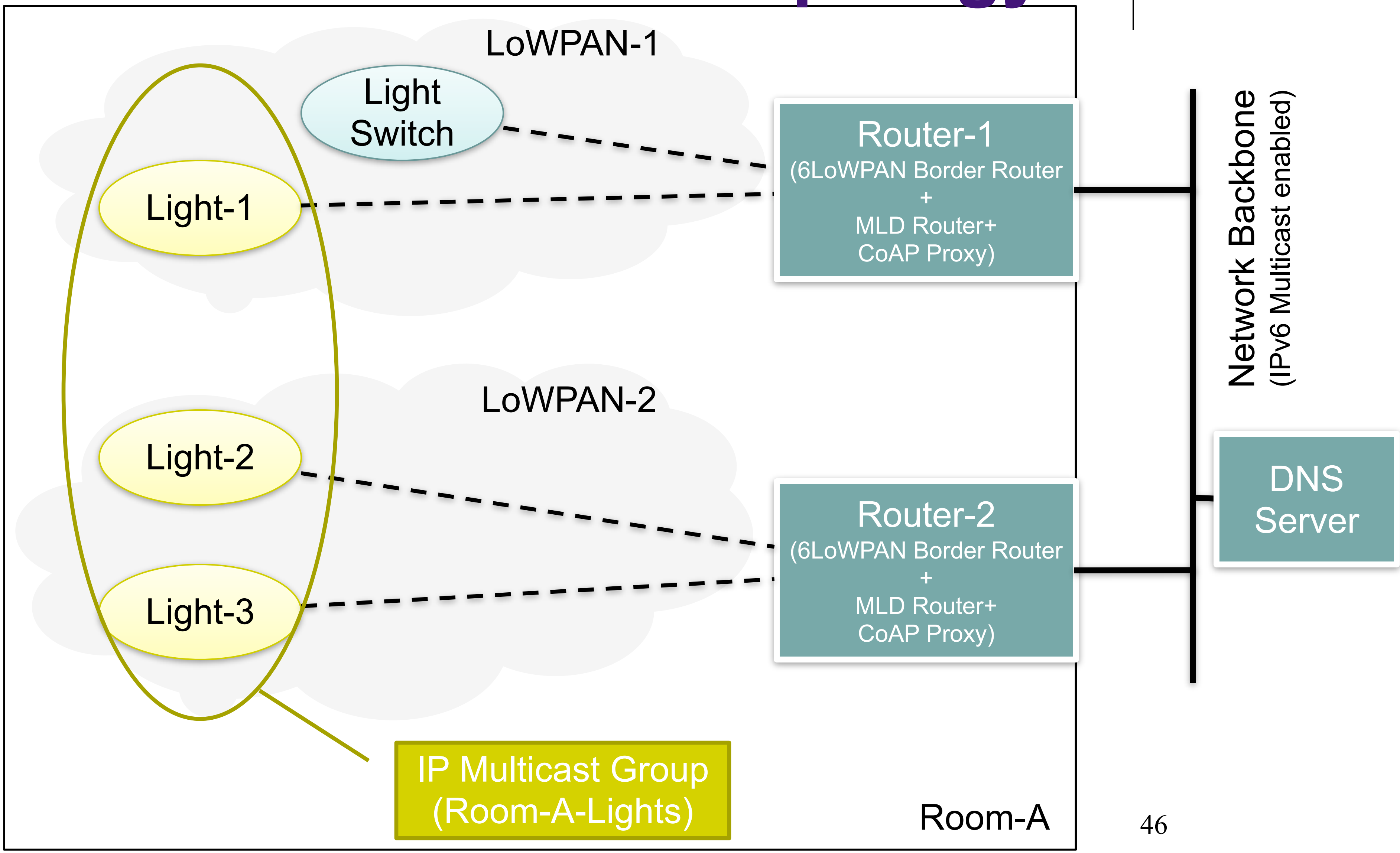
3) Content is distributed to all members of group (e.g. multicast, series of multicast, or serial unicast)

4) Optional Response

Use Case (and Example Protocol Flow)

TURNING ON LIGHTS IN A LARGE CONFERENCE ROOM

Room-A Network Topology



Turning on lights in Room-A (1/4)

Light-1 | Light-2 | Light-3 | Light switch | Router-1 (CoAP Proxy) | Router-2 (CoAP Proxy)

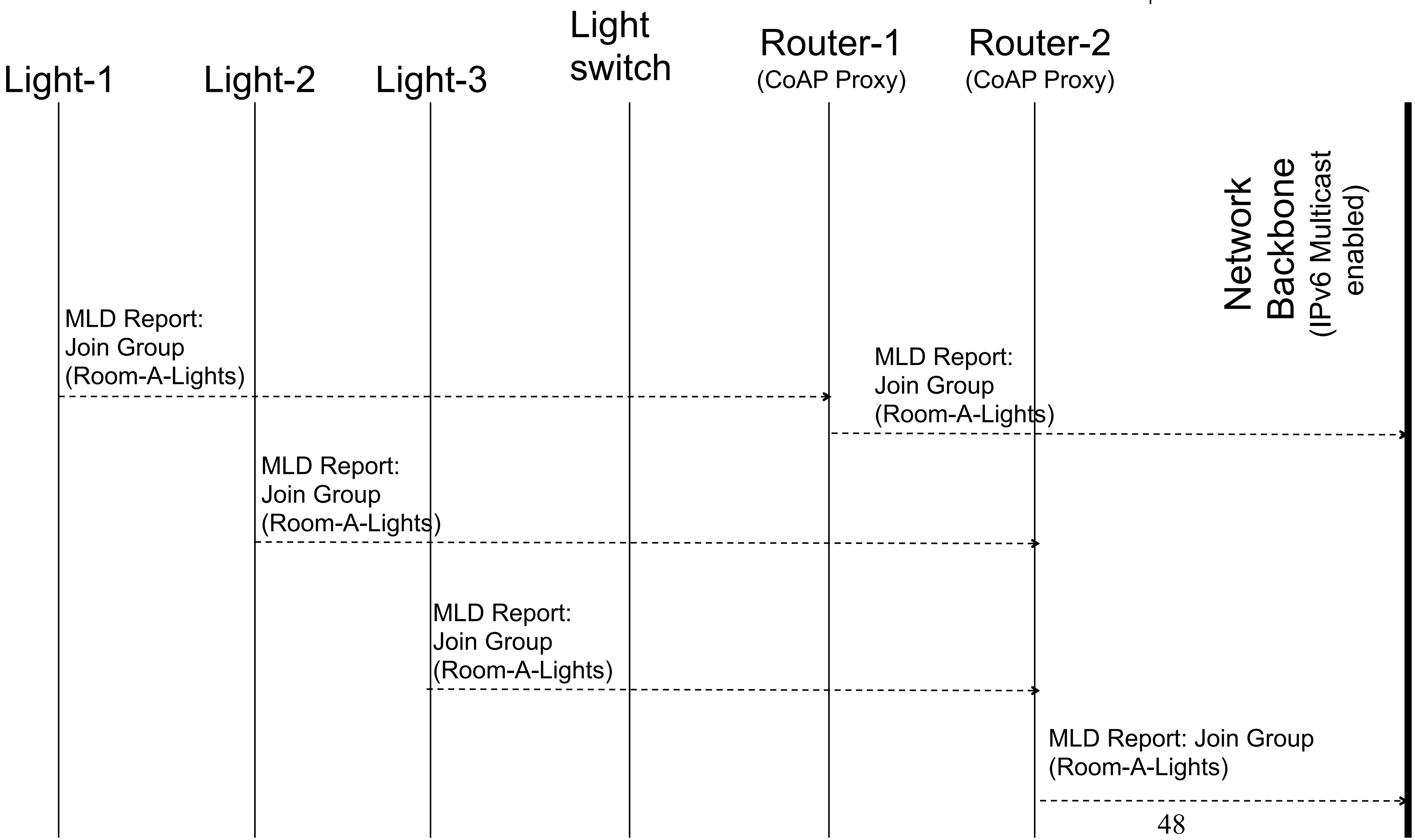
Startup phase

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

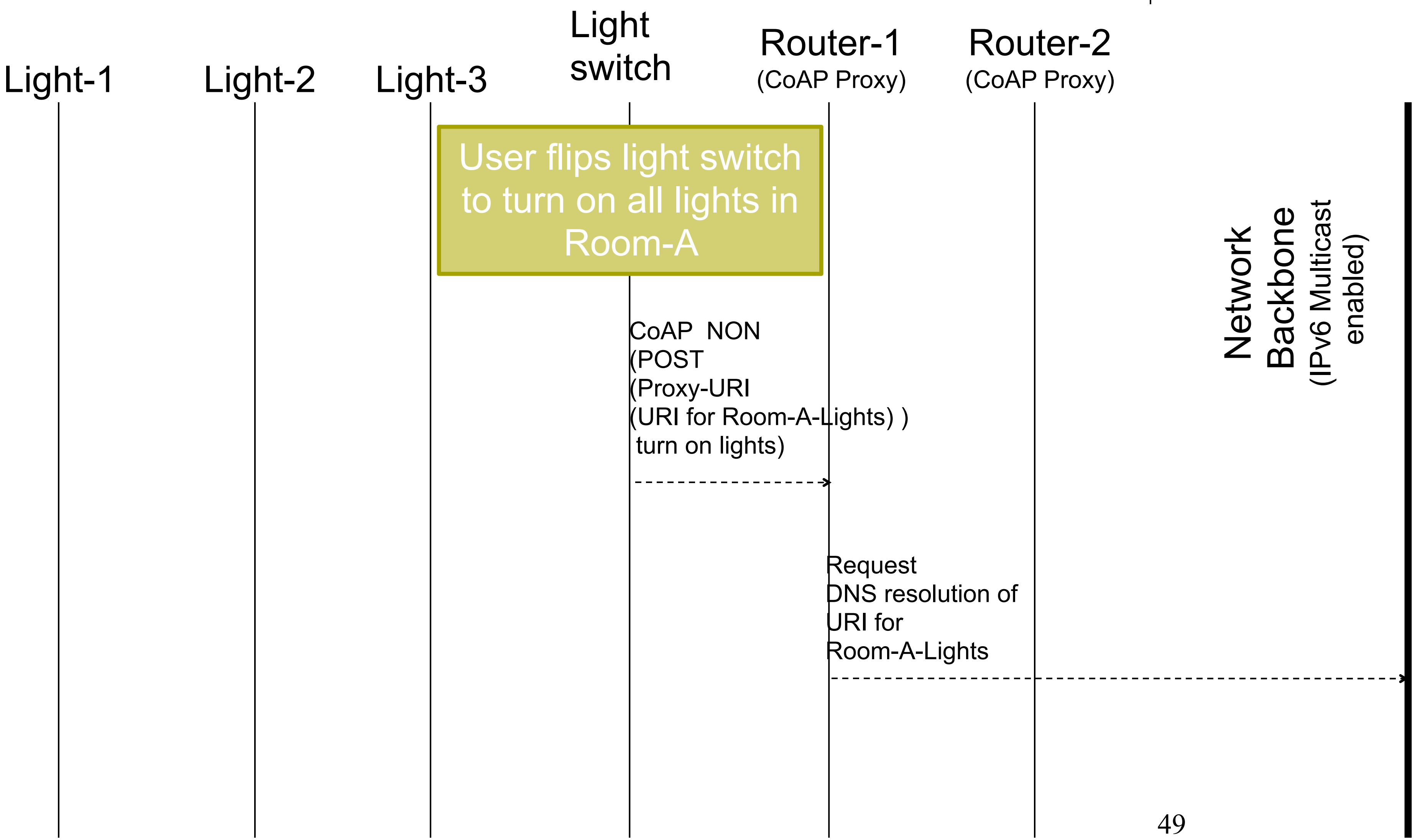
Commissioning phase (by applications)

- 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.

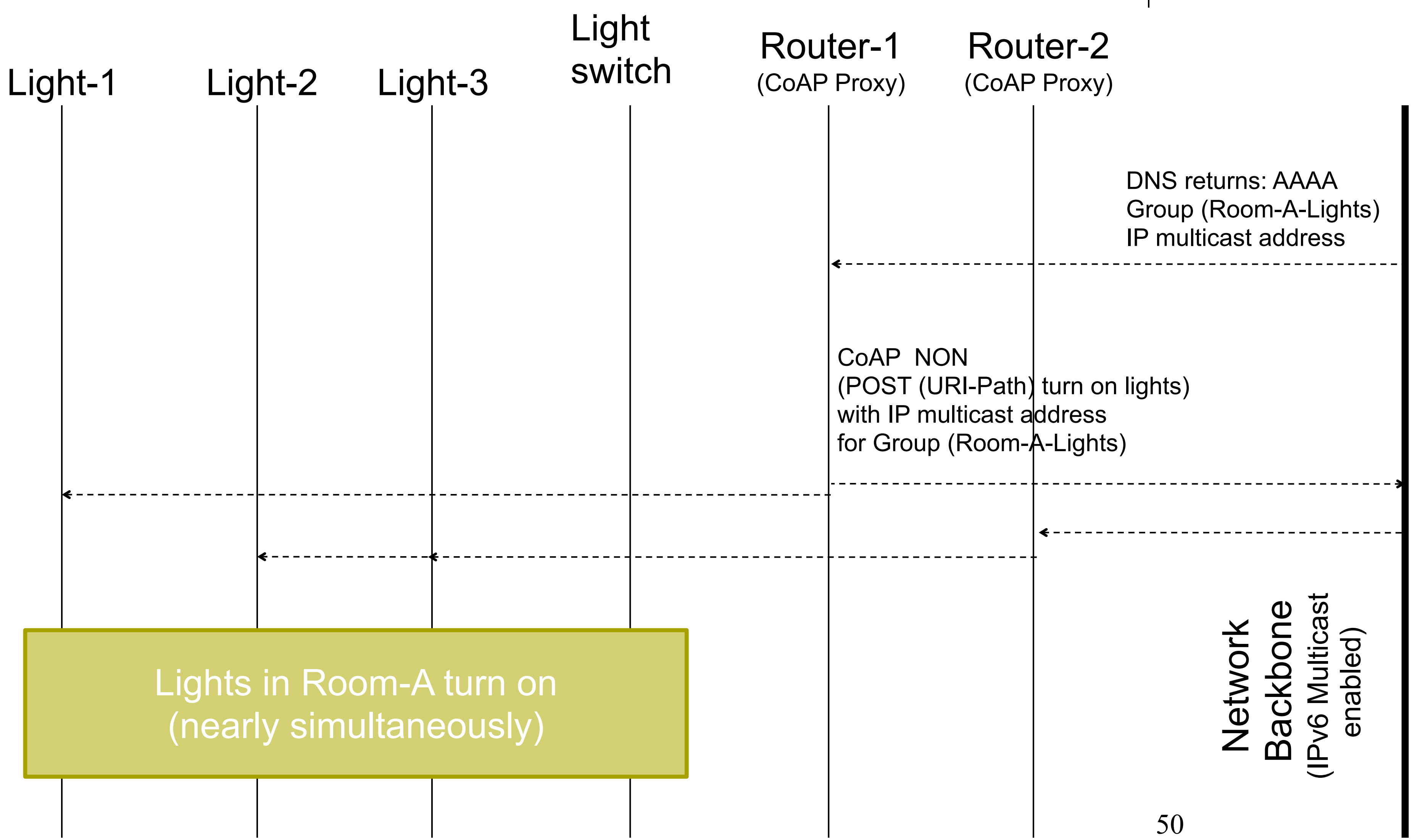
Turning on lights in Room-A (2/4)



Turning on lights in Room-A (3/4)



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



Potential Approaches for Group Communication

- There are three alternative approaches possible for CoAP group communications each with associated pros/cons:
  - IP Multicast
    - Routers must support multicast protocols
  - Overlay Multicast
    - CoAP Proxy nodes must support hybrid multicast functionality
  - CoAP Application level Group Management
    - CoAP application layer must support multicast functionality

- (See backup slides for more details - reviewed in previous IETFs)

Recommended Solution (1/2)

- We recommend that IP Multicast be adopted as the base solution for CoAP Group Communication
  - This approach requires no standards changes to the IP Multicast suite of protocols
  - It does, however, require carefully implementing pieces of IP Multicast functionality in an LLN, in a backbone network, or in both

- Implementation strategies for the following target network topologies are outlined in the I-D:
  - Single LLN topology
  - Single-LLN-with-backbone topology
  - Multiple-LLNs-with-backbone topology

Recommended Solution (2/2)

- For all network topologies that were evaluated, CoAP group communication can in principle be supported with IP Multicast, making use of existing protocols
- Also potential (but optional) optimizations were identified for an “MLD-like” or “MLD-lightweight” protocol specifically for LLNs, which would interwork with regular MLD on the backbone network
  - E.g.: A subset of MLD could be defined for an “MLD for 6LowPAN” to minimize complexity for constrained nodes
  - Reason for MLD: independence from specific LLN routing protocols for 6LNs (hosts)

What does the WG Recommend?

1. Accept the recommendation and adopt as a WG document?

2. Do more investigation?
   - We can keep updating the I-D if the WG feels there are still open issues

BACKUP

Background

- This draft is a follow up to our previous draft on "Sleeping and Multicast Considerations for CoAP" which was in a problem statement format:
  - http://tools.ietf.org/html/draft-rahman-core-sleeping-00
- During the previous CORE Webex calls, we were asked to produce satellite drafts to more precisely identify the problems and provide some initial solution proposals for:
  - Group Communications (as the more general problem of multicast) – This draft
  - Sleeping Nodes – TBD draft (but in progress)

Requirements for Group Comm (1/4)

- REQ1: Selectable Reliability:
  - At least unreliable group communication supported, but preferably reliable group communications as well if possible
- REQ2: Efficiency:
  - Delivers messages more efficiently than a “serial unicast only” solution. Also, it should provide a right balance between group data traffic and control overhead
- REQ3: Low Latency:
  - Deliver a message (preferably) as fast as possible
- REQ4: Synchrony:
  - Allows near-simultaneous modification of a resource on all devices in a group, providing to users a perceived effect of synchrony or simultaneity
  - It can be expressed as a time span “D” such that message “m” is delivered to all destinations in a time interval [t, t+D] for arbitrary “t”

Requirements for Group Comm (2/4)

- REQ5: Ordering
  - [TBD to check what use cases require in terms of message ordering especially in multi-source situations]
- REQ6: Security
  - See Backup slides for 7 security requirements (reviewed in IETF Prague)
- REQ7: Flexibility:
  - Support for one or many source(s), for dense and sparse networks, for high or low listener density, one or many group(s), and multi-group membership
- REQ8: Robust Group Management:
  - Includes functionality to join groups, leave groups, view group membership, and persistent group membership in failing node or sleeping node situations

Requirements for Group Comm (3/4)

- REQ9: Network Layer Independence
  - A solution should be specified independent from specific unicast and/or IP multicast routing protocols
  - It should support different routing protocols and implementations thereof
- REQ10: Minimal Specification Overhead
  - A group communication solution should preferably re-use existing/established (IETF) protocols that are suitable for Low Power Lossy Network (LLN) and standard backbone deployments, instead of defining new protocols from scratch
- REQ11: Minimal Implementation Overhead
  - E.G. A solution allows to re-use existing (software) components that are already present on constrained nodes such as (typical) 6LoWPAN/CoAP nodes

Requirements for Group Comm (4/4)

- REQ12: Mixed backbone/LLN Topology Support
  - A solution should work within a single LLN, and in combined LLN/backbone network topologies, including multi-LLN topologies
  - Both the senders and receivers of CoAP group messages may be attached to different network links or be part of different LLNs, possibly with routers or switches in between group members
  - In addition, different routing protocols may operate on the LLN and backbone networks. Preferably a solution also works with existing, common backbone IP infrastructure (e.g. switches or routers)
- REQ13: CoAP Proxying Support
  - A CoAP proxy can handle distribution of a message to a group on behalf of a (constrained) CoAP client
- REQ14: Suitable for operation on LLNs with constrained nodes

IP Multicast

- Concept:
  - CoAP sub-networks to be connected directly to IP multicast enabled routers (e.g. running PIM-SM [RFC4601]).
  - Sending CoAP node can directly transmit group messages by setting IP address to selected multicast IP group address
  - Receiver CoAP nodes use MLD [RFC3810] to subscribe (listen) to any messages sent to selected IP multicast group
- Pros
  - Most efficient solution since done at IP layer
  - ROLL [draft-ietf-roll-rpl-14] assumes IP multicast supported
  - CoAP-03 draft [section 4.1] assumes IP multicast supported
- Cons
  - IP multicast is not generally deployed outside of corporate LANs and a few ISPs. So we may specify IP multicast support but practically it may often not be deployed

Overlay (Proxy based) Multicast (1/2)

- Concept:
  - We define overlay multicast as one that utilizes an infrastructure based on proxies (rather than an IP router based multicast backbone) to deliver IP multicast packets to an end device
  - Since ROLL and CoAP drafts already support MLD (see pg. 4), we propose MLD Proxy [RFC3810] to be used as the overlay multicast approach
  - Specifically, the CoAP proxy node will also support Proxy MLD
  - Receiver CoAP nodes use MLD Proxy signaling to subscribe (listen) to any messages sent to selected IP multicast group
  - The CoAP (MLD) proxy node would be responsible for delivering any IP multicast message to the subscribed CoAP devices
  - Note that the CoAP (MLD) proxy need not necessarily be connected to an external multicast backbone

Overlay (Proxy based) Multicast (2/2)

- Pros
  - Ties well into existing CoAP proxy concept
- Cons
  - It is not obvious that existing MLD Proxy [RFC 3810] allows the specific scenario we are proposing. Further investigation required.

CoAP Application level Group Mgmt

- Concept:
  - Perform all group communications at the CoAP application level
  - Expand CoAP headers to allow simple group mgmt functions (Join, Leave, etc.)
  - The CoAP proxy node would be responsible for group mgmt
  - Any CoAP node that wanted to send a message to a CoAP group would first send the CoAP message to the proxy. The proxy would then explode it out to the group
- Pros
  - Functionality fully within the CoAP protocol (and CORE WG control)
  - Analogous approach as Email group management (and other Apps)
- Cons
  - Has high overhead compared to lower layer solutions

Group Resource Manipulation (1/3)

- Needed to replicate functionality of existing standards, e.g. BACnet's Alarm and Event Notification service
- Two forms of group resource manipulation should be supported:
  - Push (PUT or MPUT) as for example "turn off all lights simultaneously"
  - Pull (GET or MGET) as for example "return all the resources matching a well known URI"
- Conceptually, the result of a MGET or MPUT should be the same as if the client had unicast them serially

Group Resource Manipulation (2/3)

- Limit manipulation to idempotent methods (PUT/GET/DEL)
  - Repeat requests can then be used to increase reliability of receipt
- Requires a consistent naming and addressing scheme for groups
  - Multicast is the easy case; can use DNS to resolve FQDN in authority to multicast or unicast address
- Can a group be represented by a list of addresses as well?
  - If so, perhaps this argues for a group scheme, e.g. “coapm” to signal a proxy to do fan-out task

Group Resource Manipulation (3/3)

- Target resource must be located at same port and path for all group members
  - Suggests a need to advertise path, port or have a priori agreement

Security Considerations

- As per major comment from IETF79 (Beijing), reviewed output of:
  - IETF MSEC (Multicast Security)
    - In particular, [RFC3740], [RFC5374] and [RFC4046] are very instructive
  - IRTF SAMRG (Scalable Adaptive Multicast Research Group)
- And derived the following requirements for securing group communications in CoAP

Group Security Requirements for CoAP (1/3)

- REQ1: Group communications data encryption:
  - Important CoAP group communications shall be encrypted (using a group key) to preserve confidentiality. It shall also be possible to send CoAP group communications in the clear (i.e. unencrypted) for low value data.
- REQ2: Group communications source data authentication:
  - Important CoAP group communications shall be authenticated by verifying the source of the data (i.e. that it was generated by a given and trusted group member). It shall also be possible to send unauthenticated CoAP group communications for low value data.
- REQ3: Group communications limited data authentication:
  - Less important CoAP group communications shall be authenticated by simply verifying that it originated from one of the group members (i.e. without explicitly identifying the source node). This is a weaker requirement (but simpler to implement) than REQ2. It shall also be possible to send unauthenticated CoAP group communications for low value data.

Group Security Requirements for CoAP (2/3)

- REQ4: Group key management:
  - There shall be a secure mechanism to manage the cryptographic keys (e.g. generation and distribution) belonging to the group; the state (e.g. current membership) associated with the keys; and other security parameters.

- REQ5: Use of Multicast IPSec:
  - The CoAP protocol [I-D.ietf-core-coap] allows IPSec to be used as one option to secure CoAP. If IPSec is used at the CoAP level, then multicast IPSec [RFC5374] should be used for securing CoAP group communications.

- REQ6: Independence from underlying routing security:
  - CoAP group communication security shall not be tied to the security of underlying routing and distribution protocols such as PIM [RFC4601] and ROLL [I-D.ietf-roll-rpl]. Insecure or inappropriate routing (including multicast routing) may cause loss of data to CoAP but will not affect the authenticity or secrecy of CoAP group communications.

Group Security Requirements for CoAP (3/3)

- REQ7: Interaction with HTTPS:
  - The security scheme for CoAP group communications shall account for the fact that it may need to interact with HTTPS (Hypertext Transfer Protocol Secure) when a transaction involves a node in the general Internet (non-constrained network).

CoAP Multicast and HTTP Unicast Interworking (1/2)

```
  CoAP              CoAP            CoAP/HTTP            HTTP
 Node 1            Node 2             Proxy              Node
   |                 |                  |                  |
   |   REQUEST       |                  |                  |
   |  (Group Join)   |                  |                  |
   |---------------- |------------- >   |                  |
   |   RESPONSE      |                  |                  |
   |< -------------- |--------------    |                  |
   |                 |                  |                  |
   |                 |    REQUEST       |                  |
   |                 |  (Group Join)    |                  |
   |                 |------------- >   |                  |
   |                 |   RESPONSE       |                  |
   |                 |< ------------    |                  |
   |                 |                  |                  |
   |                 |                  |  HTTP REQUEST    |
   |                 |                  |    (URI to       |
   |                 |                  |   unicast addr)  |
   |                 |                  |< ----------------|
   |                 |                  |                  |
   |                 |              Map URI                |
   |                 |       to multicast address          |
   |                 |                  |                  |
   |   REQUEST (to multicast addr)      |                  |
   |                 |< ------------    |                  |
   |< -------------- |--------------    |                  |
   |                 |                  |                  |
   |     (optional) RESPONSE            |                  |
   |                 |------------- >   |                  |
   |---------------- |-------------->   |                  |
   |                 |                  |  HTTP RESPONSE   |
   |                 |                  |---------------- >|
   |                 |                  |                  |
```

CoAP Multicast and HTTP Unicast Interworking (2/2)

- Proxy node needs to have the following functionalities to interwork CoAP/UDP (multicast) and HTTP/TCP (unicast):
  - Incoming HTTP Request will carry a URI (with HTTP scheme)
  - At the proxy node, the URI will then be again resolved (with CoAP scheme) to an IP multicast. This may be accomplished, for example, by using DNS-SD
  - The proxy node will then multicast the CoAP Request to the appropriate nodes
- CoAP proxy can be considered to be a "non-transparent" proxy according to [RFC2616]:
  - Specifically, [RFC2616] states that a "non-transparent proxy is a proxy that modifies the request or response in order to provide some added service to the user agent, such as group annotation services, media type transformation, protocol reduction or anonymity filtering."

Use case optimizations

For improved latency e.g.

1. DNS AAAA records cached in 6LBR
2. Light switch sends CoAP multicast directly (non-proxied) to group RM-A
3. Light switch is commissioned directly with multicast IP address of group RM-A, not URI

Hallway discussions on multicast

- **Yesterday, during the cross-layer discussion meeting, it seemed:**
- **It will be hard to get good multicast forwarding out of RPL environments**
  - **requires storing mode**
    - (current implementer focus is on non-storing mode)
  - **efficient multicast would need some form of CDS/MPR**
  - **trickle-multicast aims at reaching everyone, slowly**
- **There is probably a lot of work necessary to make multicast work reliably in a backbone-connected system of RPL-routed 6LoWPANs etc.**

82nd IETF: core WG Agenda

15:20	Introduction, Agenda, Status	Chairs (10)
15:30	1 – core CoAP	ZS (90)
17:00	1 – block, observe	CB (20)
17:20 retire to Friday, 11:20	Intro	Chairs (05)
11:25	1 – core CoAP: artificial limitations	CB (10)
11:35	Group Communication	AR (10)
11:43	Naming, Data Formats	JA (10)
11:57	Discovery	ZC MI ZS
12:27	Using CoAP/Naming	KL (8)
12:35	HTTP Mapping	AR (5)
12:40	security	SLK AY BS
13:10	new: CoAP-over-X	KL (10)
13:20	next steps	Chairs (10)
13:30 retire		

Device URN

urn:dev:ow:10e 2073a 01080063

Jari Arkko, Cullen Jennings, Zach Shelby

What's the Problem?

1. Google for ”XML sensor data format”
2. Take the first search result
3. Go to the first example on the page

The CC128 message structure is slightly compressed compa

free-form text fields

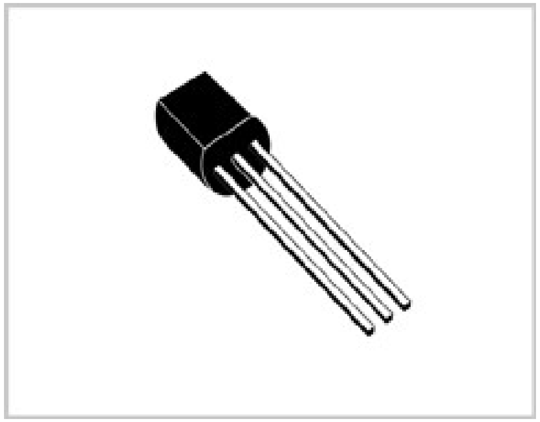
```
<msg>                                start of message
  <src>CC128-v0.11</src>             source and software version
  <dsb>00089</dsb>                   days since birth, ie days run
  <time>13:02:39</time>              24 hour clock time as displayed
  <tmpr>18.7</tmpr>                  temperature as displayed
  <sensor>1</sensor>                 Appliance Number as displayed
  <id>01234</id>                     radio ID received from the sensor
  <type>1</type>                     sensor Type, "1" = electricity
  <ch1>                              sensor channel
    <watts>00345</watts>             data and units
  </ch1>
```

Text-Based vs. Uniform Identifiers

- Cannot make any use of the text identifiers beyond exact match
- Text identifiers do not have clearly defined scope or uniqueness properties
- Uniform, formally defined identifiers can be passed around more easily:
  - They are self-describing
  - Merging data from different sources easier
  - No coordination needed across types
- Conclusion 1: use URNs or URIs as identifiers

Identifier Types

- Semantics-based ”sensor for the oven”
- Name-based ”my_sensor_3”
- Location-based ”coordinates X,Y”
- Address-based ”http://[2001:db8::1]”
- Device ID-based (”mac=...” or ”serial=...”)



How do you configure this device to send a name or location?

Conclusion 2: Device IDs are attractive for many deployment cases – e.g., identifying specific devices in sensor data streams, storage servers and equipment inventory applications. Names are obviously needed too, but can exist at higher layers

The Specification for ”dev” URNs

urn:dev:mac:0024befffe804ff1 (my laptop's MAC address)

- Device identifiers based on EUI-48/64 MAC addresses
  - Similar to UUIDs but requires no real-time clocks, stable storage, and has easier process on the manufacturing side
- Device identifiers based on 64-bit 1-Wire addresses
- Device identifiers based on cryptographic identifiers – related to the security discussion from yesterday
- Extension rules for new types

SenML

draft-jennings-senml


```
{"e":[
        {"v":"23.5", "t": "0"},
        {"v":"23.4", "t": "10"},
        {"v":"23.4", "t": "20"},
        {"v":"23.3", "t": "30"},
        {"v":"23.2", "t": "40"},
        {"v":"23.0", "t": "50"},
        {"v":"22.0", "t": "60"},
        {"v":"19.3", "t": "70"},
        {"v":"17.21", "t": "80"},
        {"v":"17.03", "t": "90"},
        {"v":"16.9", "t": "100"}
      ],
 "bt":"1276020076",
 "bn":"urn:dev:ow:10e2073a01080063"
}
```


Cullen Jennings

Zach Shelby

Jari Arkko

Why?

- Smart objects need common data format(s)
- JSON is an easy, relatively compact format
- Properly designed base format helps use a generic data container for typical smart object applications – no need to design a scheme just to represent temperature measurements
- Right design helps keep size down even on textual format
- JSON, XML, EXI mappings

```
root@weather:/home/jar/OneWire/History# ls -l 26.*
-rw-r--r-- 1 root root 50436604 2011-11-17 21:44 26.2B4DF5000000.history.dat
-rw-r--r-- 1 root root 75752642 2011-11-17 21:44 26.2D5FE7000000.history.dat
-rw-r--r-- 1 root root 50438850 2011-11-17 21:44 26.4437F5000000.history.dat
-rw-r--r-- 1 root root 95495758 2011-11-17 21:44 26.80A3CD000000.history.dat
root@weather:/home/jar/OneWire/History#
```

82nd IETF: core WG Agenda

15:20	Introduction, Agenda, Status	Chairs (10)
15:30	1 – core CoAP	ZS (90)
17:00	1 – block, observe	CB (20)
17:20 retire to Friday, 11:20	Intro	Chairs (05)
11:25	1 – core CoAP: artificial limitations	CB (10)
11:35	Group Communication	AR (10)
11:43	Naming, Data Formats	JA (10)
11:57	Discovery	ZC MI ZS
12:27	Using CoAP/Naming	KL (8)
12:35	HTTP Mapping	AR (5)
12:40	security	SLK AY BS
13:10	new: CoAP-over-X	KL (10)
13:20	next steps	Chairs (10)
13:30 retire		

CoAP-HTTP Proxy Discovery and Energy-aware Resource Discovery

draft-cao-core-pd-00

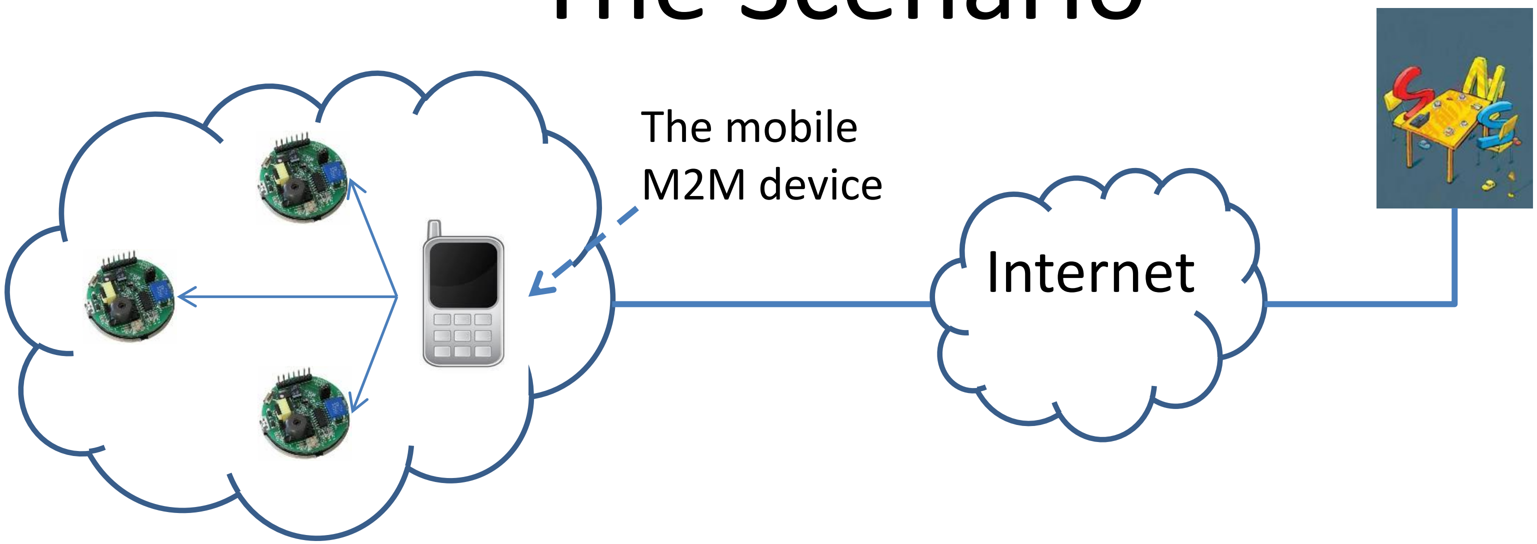
draft-ma-core-dhcp-pd

draft-he-proxy-discovery-coap-00.txt

The Problem

- CoAP-HTTP is needed and specified
  - draft-ietf-core-coap
  - draft-ietf-castellani-core-http-mapping
- How a HTTP client discovers a forward proxy to the CoAP server
  - DNS based discovery is adequate more than enough, see draft-vanderstok-core-bc-05
- How a CoAP client discovers a forward proxy to the HTTP server
  - Main focus of these two drafts
- More general DISCOVERY solutions:
  - draft-brandt-coap-subnet-discovery
  - draft-bormann-core-simple-server-discovery
- Design space
  - Static configuration: DHCP based
  - Dynamic discovery: Link-format based

The Scenario



- The sensors are static, running CoAP as a client, and wants to report information to the SNS website
- The “mobile M2M devices” are nomadic and can serve as the CoAP-HTTP proxy
- The sensor wants to discover who is nearby and can shepherd message to the Web

Proposal #1: Link-format Discovery

Multicast CoAP GET

```
End-point                                    Multicast address      Proxy
   |                                                 |                 |
   |  -- GET /.well-known/core?rt=core-pd -->|                 |
   |                                                 |                 |
   |                                                 |                 |
   |<----------- 2.05 Content ; rt="core-pd" ----------------- |
   |                                                                   |
   |----------------------    GET /temp/   ---------------------->|

   Req: GET coap://[ff02::1]/.well-known/core?rt=core-pd

   Res: 2.05 Content
   fe80::ff; rt="core-pd";
```

The Proxy Replies

The sensor read the proxy address from the packet

Proposal #2: DHCP option for proxy discovery on IPv4 sensor node

```
Code  Len   Address 1              Address 2
+-----+-----+-----+-----+-----+-----+-----+-----+--
|  x  |  n  |  a1 |  a2 |  a3 |  a4 |  a1 |  a2 |  ...
+-----+-----+-----+-----+-----+-----+-----+-----+--

                CoAP proxy/rd option
```

Other consideration: proxy discovery for IPv6 sensor (Thanks to Tomas's comment)

1. DHCPv6 option?
2. the ABRO option of 6LoWPAN-ND as the hint

Energy-aware Resource Discovery

draft-he-proxy-discovery-coap-00.txt

Process in draft-shelby-core-resource-directory-00: Discovery and Register

```
End-point                                              RD
    |                                                   |
    | ----- GET /.well-known/core?rt=core-rd ------>    |
    |                                                   |
    |                                                   |
    | <---- 2.05 Content "</rd>; rt="core-rd" ------    |
    |                                                   |
```

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

Discovery

```
End-point                                              RD
    |                                                   |
    | --- POST /rd "</sensors..." ---------------->     |
    |                                                   |
    |                                                   |
    | <-- 2.01 Created Location: /rd/node1 ---------    |
    |                                                   |
```

Req: POST coap://rd.example.org/rd?n=node1&lt=1024
Etag: 0x3f
Payload:
</sensors/temp>;ct=41;rt="TemperatureC";if="sensor",
</sensors/light>;ct=41;rt="LightLux";if="sensor"
Res: 2.01 Created
Location: /rd/node1

Register

Proposal:

1. POST to .well-known/core?rt=core-rd URI for direct register
2. Default /rd for resource directory record

```
End-point                                                    Neighbor
    |                                                      (as a proxy)
    | --- POST /.well-known/core?rt=core-rd------->            |
    |                                                          |
    |                                                          |
    | <-- 2.01 Created Location: /rd/node1 ---------           |
    |                                                          |
```

Req: POST coap://[ff02::1]/.well-known/core?rt=core-rd
Payload:
</sensors/temp>;ct=41;rt="TemperatureC";if="sensor",

Res: 2.01 Created
Location: /rd/node1

Pros: one round-trip to save energy

Cons:

- Lifetime mandatory? (according to Salvatore's comment)
- All on the link should parse the POST request

Constrained Device Configuration

draft-nieminen-core-service-discovery

IETF 82

markus.isomaki@nokia.com

Constrained Device Configuration

- Many devices are not constrained only by communications and processing, but also input/output
- We know how to auto-configure them for IP connectivity and other “local” things
    - RA/RS, DHCP, multicast, anycast, mDNS, …
- But they may also need service provider and user account configuration before they can do anything useful
    - CoAP/HTTP server, user credentials
    - Compare to bare bones e-mail, XMPP or SIP account configuration
- If they are really constrained, they can’t even offer a web form for setup

- Can we help the poor user to setup her new gadget?

Potential tools – CoAP and a helper device

- Assumptions
  - The constrained device supports at least CoAP
  - The user has a PC, tablet or smartphone that can communicate with the device over IP

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

2.05 Content

PUT /config
{“s”:”coap.example.com”, “u”:”username”, ”p”:”passwd”}

- Or vice versa, the constrained device can start the discovery and do a GET

What’s in it for IETF

- Is this a relevant problem to solve in a general way?
- Is CoAP and resource discovery a reasonable starting point?
    - Define a new resource type
- Could we agree on the basic configuration data model?
    - Just enough to allow the device to connect to a CoAP server with user credentials
    - Text, JSON, SenML, …

CoRE Resource Directory

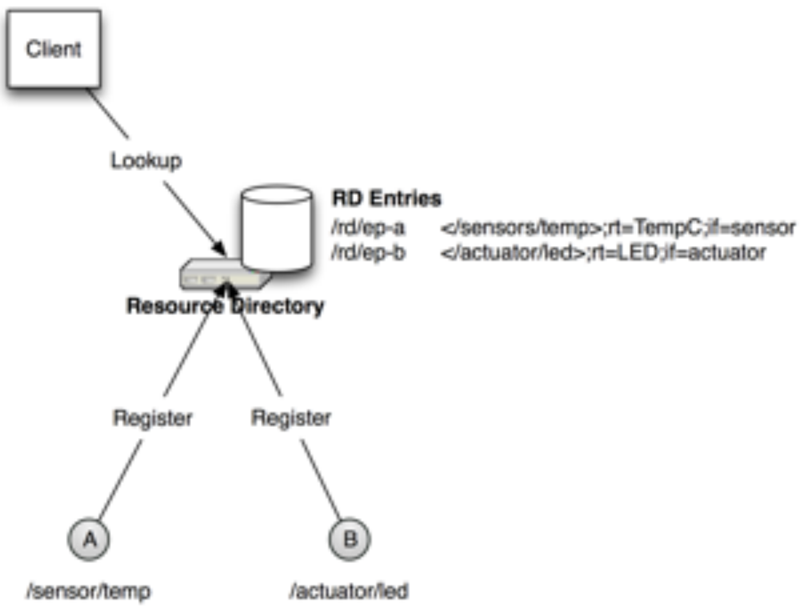
draft-shelby-core-resource-directory-02

Z. Shelby, S. Krco

CoRE WG, IETF-82 Taipei

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
- Used in CoRE and IPSO Alliance interop events
- Currently being deployed
  - in commercial products
  - by telecom operators
  - in EU project trials



Try it out yourself

- I have made a simple test Resource Directory available
  - Originally developed for a recent IPSO interop event
  - Supports IPv4 and IPv6
- CoAP interface as per resource-directory-01:
  - coap://interop.ams.sensinode.com/rd
- HTTP (HTML) interface also available:
  - http://interop.ams.sensinode.com
- Go ahead and POST your CoAP resources!
- Test CoAP server also available at
  - coap://interop.ams.sensinode.com:8000

Changes since -01

- Technical:
  - Changed the inclusion of an Etag in registration or update to a MAY
  - Added the concept of an RD domain and a registration parameter for it
  - Recommended the Location returned from a registration to be stable, allowing for end-point and domain information to be changed during updates
  - Changed the lookup interface to accept end-point and domain as query string parameters to control the scope of a lookup
- Editorial:
  - Added a terminology section

Known Issues/Ideas

- The lookup interface SHOULD support Observation
  - For the purpose of synchronization
- The lookup interface could be more sophisticated
  - Lookup the end-points in a domain (and filter them)
  - Lookup the domains in an RD
- The ins= attribute could be used to include a unique device identifier in a registration POST
  - Use draft-arkko-core-dev-urn-01 format
  - The h= attribute should then be changed to ep=
- More use case examples are needed
- Looking forward to more feedback

82^{nd} IETF: core WG Agenda

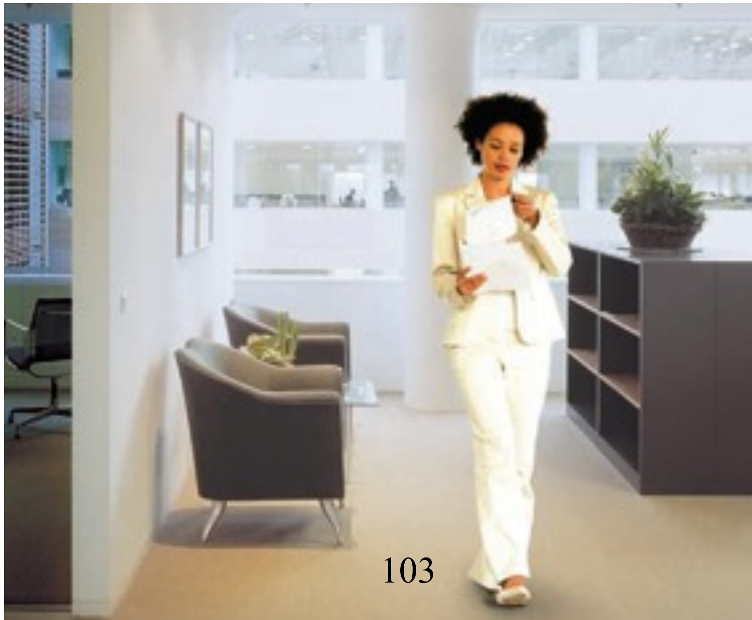
15:20	Introduction, Agenda, Status	Chairs (10)
15:30	1 – core CoAP	ZS (90)
17:00	1 – block, observe	CB (20)
17:20 retire to Friday, 11:20	Intro	Chairs (05)
11:25	1 – core CoAP: artificial limitations	CB (10)
11:35	Group Communication	AR (10)
11:43	Naming, Data Formats	JA (10)
11:57	Discovery	ZC MI ZS
12:27	Using CoAP/Naming	KL (8)
12:35	HTTP Mapping	AR (5)
12:40	security	SLK AY BS
13:10	new: CoAP-over-X	KL (10)
13:20	next steps	Chairs (10)
13:30 retire		

Naming & Discovery for Building Control

draft-vanderstok-core-bc-05

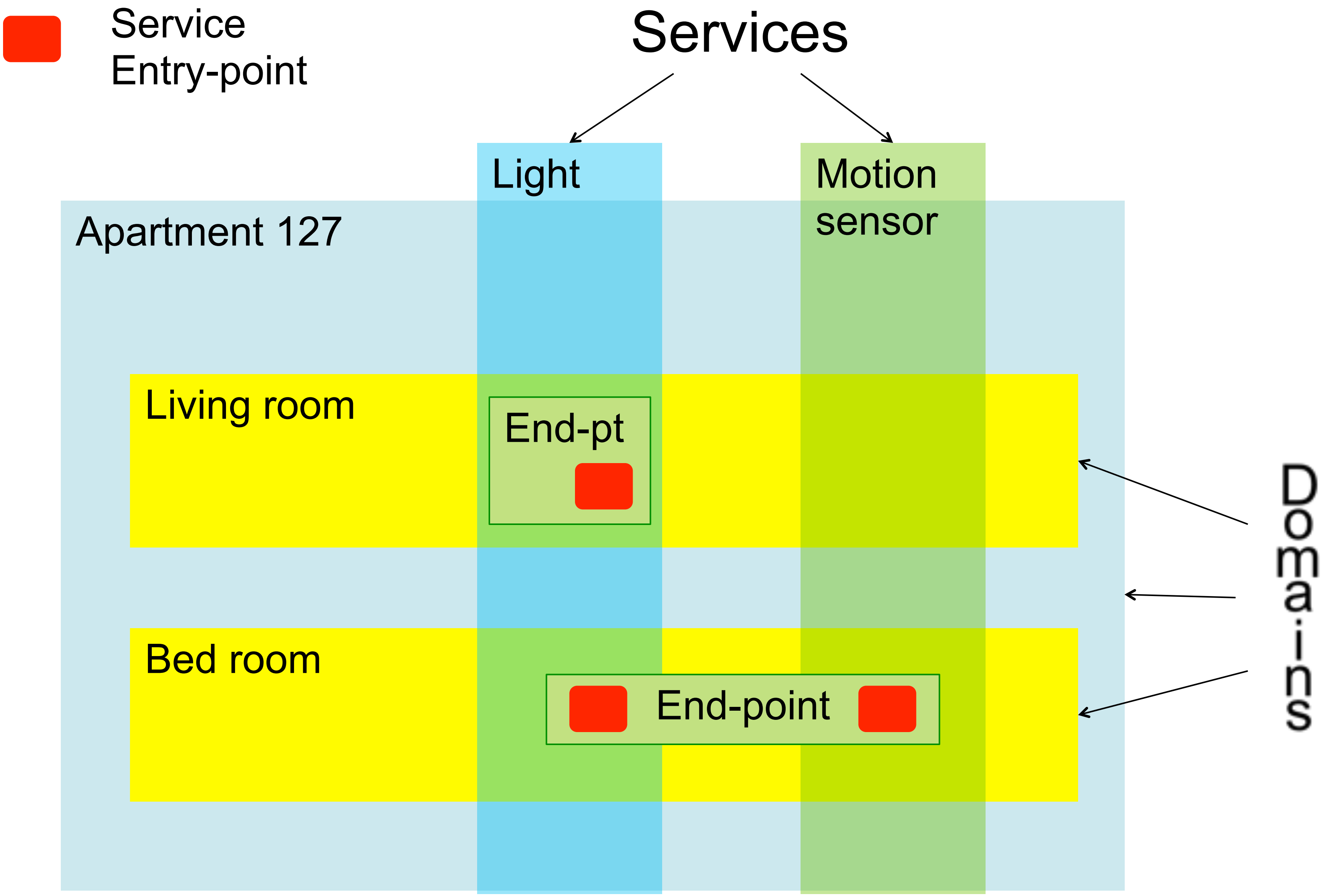
Peter van der Stok;
Anders Brandt;
Kerry Lynn

IETF-82, CoRE WG
18 November 2011

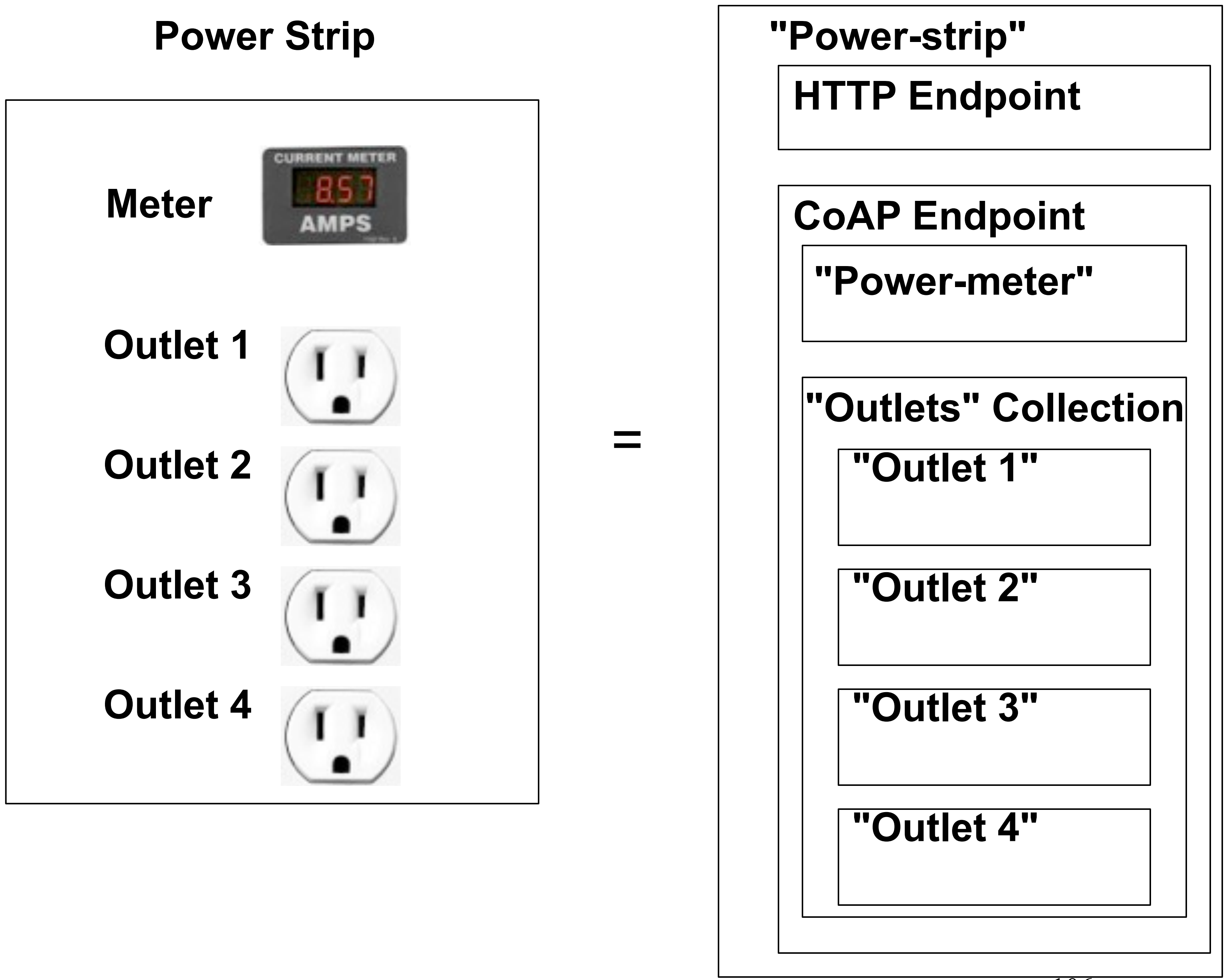


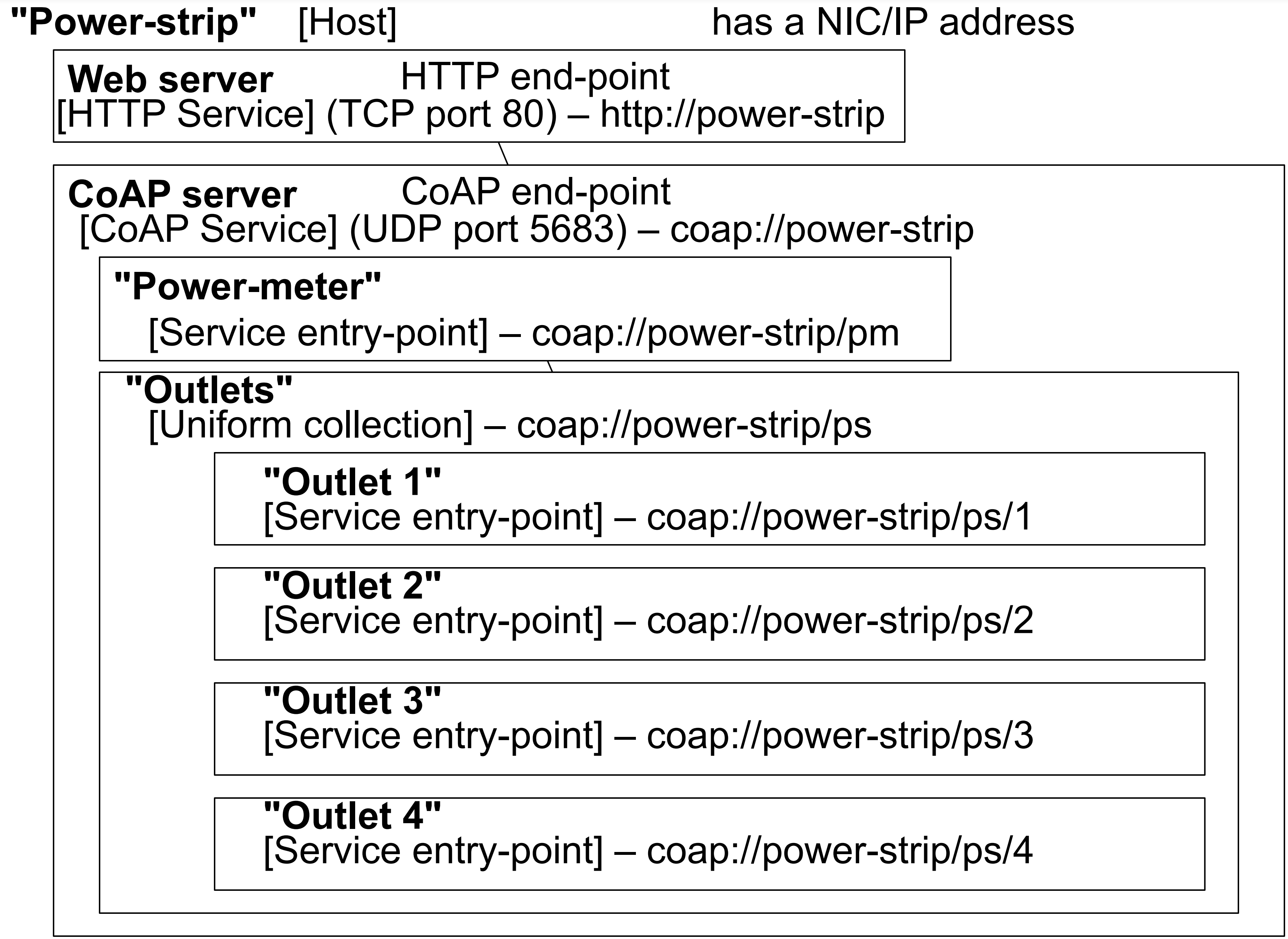
What Objects Must We Name/Resolve/Discover?

Preferred Term	Alternatives	Meaning
Host	Device, Node	Physical object bound to at least one IP address (A, AAAA) and optionally a host ID
Multicast Group	Group	A set of hosts listening on a common multicast address
End-point	Server, Service	{protocol, host, port} tuple, where *host* may name a group and *port* may have a default value (SRV)
Entry-point	Link, REST Interface, WEB Application	End-point plus a path (TXT)
Entry-point Collection	Uniform Collection	Parent path for a set of identical subordinate end-points
Domain	Zone	A logical set of end-points; used to scope a discovery query



Discovery of services within domains


Power Strip
Meter
CURRENT METER
8.57
AMPS
Outlet 1
Outlet 2
Outlet 3
Outlet 4
=
"Power-strip"
HTTP Endpoint
CoAP Endpoint
"Power-meter"
"Outlets" Collection
"Outlet 1"
"Outlet 2"
"Outlet 3"
"Outlet 4"


"Power-strip" [Host] has a NIC/IP address
Web server HTTP end-point
[HTTP Service] (TCP port 80) – http://power-strip
CoAP server CoAP end-point
[CoAP Service] (UDP port 5683) – coap://power-strip
"Power-meter"
[Service entry-point] – coap://power-strip/pm
"Outlets"
[Uniform collection] – coap://power-strip/ps
"Outlet 1"
[Service entry-point] – coap://power-strip/ps/1
"Outlet 2"
[Service entry-point] – coap://power-strip/ps/2
"Outlet 3"
[Service entry-point] – coap://power-strip/ps/3
"Outlet 4"
[Service entry-point] – coap://power-strip/ps/4

Link-format to DNS-SD mapping

Link Format	DNS-SD
Resource Instance (ins=)	{Instance}
Resource Type (rt=)	{ServiceType}
<uri>	TXT path=
Interface Description (if=)	TXT if=
Attribute (xxx=)	TXT xxx=

Things decided by the mapping entity:

- Domain name (the DNS server where the records are created)
- Host name (if it doesn't exist already)
- txtver=*n* (TXT record version)

82nd IETF: core WG Agenda

15:20	Introduction, Agenda, Status	Chairs (10)
15:30	1 – core CoAP	ZS (90)
17:00	1 – block, observe	CB (20)
17:20 retire to Friday, 11:20	Intro	Chairs (05)
11:25	1 – core CoAP: artificial limitations	CB (10)
11:35	Group Communication	AR (10)
11:43	Naming, Data Formats	JA (10)
11:57	Discovery	ZC MI ZS
12:27	Using CoAP/Naming	KL (8)
12:35	HTTP Mapping	AR (5)
12:40	security	SLK AY BS
13:10	new: CoAP-over-X	KL (10)
13:20	next steps	Chairs (10)
13:30 retire		

Best practices for HTTP-CoAP mapping implementation

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

Introduction

- The I-D provides a base reference documentation for HTTP-CoAP (HC) proxy implementers
- It details deployment options, discusses possible approaches for URI mapping, and provides useful considerations related to protocol translation

Main Updates

- Added details for HTTP -> CoAP mapping details for the following features:
  - Cache Refresh via Observe
  - Use of CoAP blockwise transfer
  - Multicast response caching

- New section on CoAP -> HTTP mapping
  - As per discussions with Klaus, adapted sections from his expired draft (draft-hartke-core-coap-http-00)

Cache Refresh Via Observe (1/2)

- Cache Refresh via Observe (for HTTP -> CoAP mapping):
  - There are cases where using CoAP observe protocol [I-D.ietf-core-observe] to handle proxy cache refresh may be preferable to the validation mechanism based on ETag's defined [I-D.ietf-core-coap]
  - For example:
    - Sleeping nodes, possibly showing high variance in requests' distribution, would greatly benefit from a server driven cache update mechanism
    - Other expected candidates would be the crowded or very low throughput networks, where minimization of the total number of exchanged messages is a major goal
  - The I-D proposes a practical evaluation method to decide whether the refresh of a cached resource R is more efficiently handled via ETag validation or by establishing an observation on R

Cache Refresh Via Observe (2/2)

- Representative Algorithm:
  - Let:
    - T_R be the mean time between two client requests to resource R
    - F_R be the freshness lifetime of R
    - M_R be the total number of messages exchanged by the cache towards resource R in order to validate its freshness
  - Assuming a negligible initial cost for establishing the observation relationship (one only message), an observation on R lessens M_R (i.e. it's a better cache update choice then using ETag validation) iff T_R < 2*F_R
  - Or equivalently, iff the mean arrival time of requests for resource R is greater than half the refresh rate of R

CoAP Blockwise Transfer (1/2)

- Use of CoAP blockwise transfer (for HTTP -> CoAP mapping):
  - An HC proxy SHOULD support CoAP blockwise transfers [I-D.ietf-core-block] to allow transport of large CoAP payloads while avoiding link-layer fragmentation in LLNs, and to cope with small datagram buffers in CoAP end-points as described in [I-D.ietf-core-coap]
  - For improved latency an HC proxy MAY initiate a blockwise CoAP request triggered by an incoming HTTP request even when the HTTP request message has not yet been fully received, but enough data has been received to send one or more data blocks to a CoAP server already

CoAP Blockwise Transfer (2/2)

- Representative Algorithm:
  - An HC proxy SHOULD attempt to use blockwise transfer when sending a CoAP PUT or POST request message that is larger than BLOCKWISE_THRESHOLD
  - The value of BLOCKWISE_THRESHOLD is implementation-specific. For example it may be:
    - set by an administrator
    - preset to a known or typical UDP datagram buffer size for CoAP end-points
    - to N times the size of a link-layer frame where e.g. N=5
    - preset to a known IP MTU value
    - or set to a known Path MTU value

Multicast Response Caching

- Multicast response caching (for HTTP -> CoAP mapping):
  - A multicast CoAP request SHOULD be sent by a HC proxy for each incoming request addressed to a multicast group. Caching of multicast responses is still a valuable goal to pursue reduce network congestion, battery consumption and response latency
  - Caching of multicast GET responses MAY be implemented by adopting some technique that takes into account either knowledge about dynamic characteristics of group membership (occurrence or frequency of group changes) or even better its full knowledge (list of nodes currently part of the group)
  - When using a technique exploiting this knowledge, valid cached responses SHOULD be served from cache

CoAP -> HTTP Mapping

- CoAP -> HTTP mapping:
  - The I-D provides guidance on:
    - Placement and Deployment
    - Basic mapping
      - Payloads and Media Types
      - Max-Age and ETag Options
      - Use of CoAP blockwise transfer
      - HTTP Status Codes 1xx and 3xx

Next Steps

- Any technical comments from the WG?
  - Are all issues covered?
- Does the WG recommend adoption?

Backup

Cross-protocol proxies taxonomy

- Forward
  - It is explicitly known by the client
- Reverse
  - Acts as if it was the origin server
  - It knows explicitly the servers that is proxying
- Interception [RFC3040]
  - Receives requests through network interception

Cross-protocol URI

- Protocol-aware
  - Client uses the scheme specific to the protocol
    - **Example**: An HTTP client accesses coap://node.something.net/foo directly
- Protocol-agnostic
  - Client uses its natively supported scheme
    - **Example**: An HTTP client accesses coap://node.something.net/foo at an http: URI
      - The client does not even need to know the coap: URI
  - Requires cross-protocol URI mapping

URI mapping

- It is a mechanism to map a URI across two different scheme domains
  - Example: coap://node.something.net/foo is mapped to http://something.net/node/foo
- Could be complex in general
  - **Static**: the mapping does NOT change over time
  - **Dynamic**: the mapping can change over time

URI mapping examples

- Homogeneous
  - Only the scheme part of the URI changes, authority and path stay the same
    - **Example**: coap://node.something.net/foo is mapped to http://node.something.net/foo
    - Interception proxy deployments MUST use this mapping
- Embedded
  - All but the scheme part of the URI is embedded in the mapped URI
    - **Example**: coap://node.something.net/foo is mapped to http://example.com/node.something.net/foo
    - Reduces mapping complexity in reverse proxy deployments

Dynamic URI mapping (TODO)

- Dynamic URI mappings can change over time
- Useful for more complex deployments to perform various functions
  - Load-balancing
  - Handle dynamic node topology

HTTP-CoAP caching and congestion

- An HTTP-CoAP (HC) proxy using caching reduces load to CoAP servers
  – e.g. avoiding duplicate requests
- Observe relationship can be established towards "popular" resources
  – See draft-ietf-core-observe
- HC proxy may apply aggregate congestion control towards the same constrained network
  – See draft-eggert-core-congestion-control

HTTP-CoAP v4/v6 use case

DNS A record for node.coap.foo.com points to P

Or P is forward

HTTP unicast --> CoAP multicast

- Identification and mapping
  - The HC proxy understands whether an URI identifies a multicast resource
  - Maps the request to the relevant multicast group
  - The mapping depends on the multicast communication technology in use
    - see draft-rahman-core-groupcomm

HTTP unicast --> CoAP multicast (cont.)

- Request handling
  - Involves the following tasks
    - Distributing the request
    - Collecting the responses
    - Timeout handling
    - Responses aggregation and delivery
  - Some tasks depend on the multicast communication technology in use

HTTP unicast --> CoAP multicast (cont.)

Security considerations

- **Availability:**
  - **Risk**: Multicast amplification attacks
  - **Countermeasure**: Only known/authorized clients may access multicast resources

  - **Risk**: An high number of subscriptions can cause resource exhaustion
  - **Countermeasure**: Limit the number of concurrent subscription requests

Security considerations (cont.)

- **Integrity**
  - **Risk**: Cache poisoning on the CoAP side by an evil mote spoofing the response (feasible when using NoSec or even SharedKey).
  - **Countermeasure**: Use MultiKey with 1:1 identity binding, or SharedKey with procedurally secure mote crypto enrollment.

Security considerations (cont.)

- **Confidentiality**
  - A resource requested via a secure channel by the source SHOULD be mapped to a secure request (if possible) or rejected.

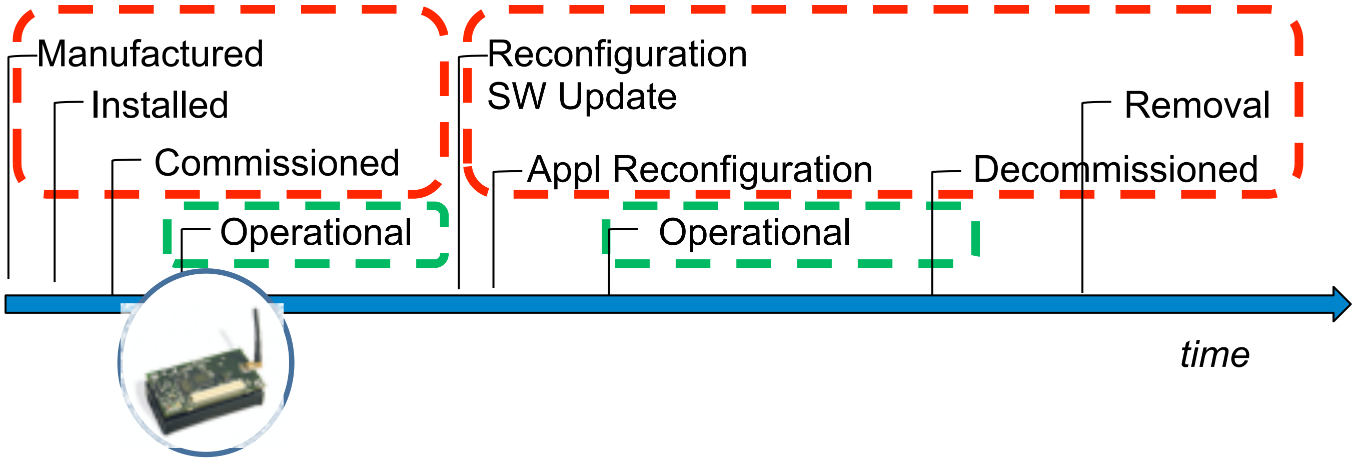
82nd IETF: core WG Agenda

15:20	Introduction, Agenda, Status	Chairs (10)
15:30	1 – core CoAP	ZS (90)
17:00	1 – block, observe	CB (20)
17:20 retire to Friday, 11:20	Intro	Chairs (05)
11:25	1 – core CoAP: artificial limitations	CB (10)
11:35	Group Communication	AR (10)
11:43	Naming, Data Formats	JA (10)
11:57	Discovery	ZC MI ZS
12:27	Using CoAP/Naming	KL (8)
12:35	HTTP Mapping	AR (5)
12:40	security	SLK AY BS
13:10	new: CoAP-over-X	KL (10)
13:20	next steps	Chairs (10)
13:30 retire		

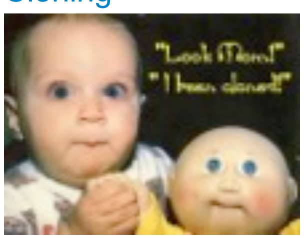
Security Considerations in the IP-based Internet of Things

Sye-Loong Keoh,
Oscar Garcia-Morchon, Sandeep S. Kumar, René Hummen and Rene Struik
Philips Research Europe, RWTH Aachen, Struik Security Consultancy

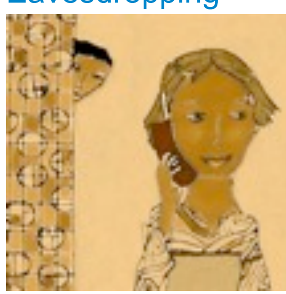
Lifecycle of a *Thing* and Threat Analysis



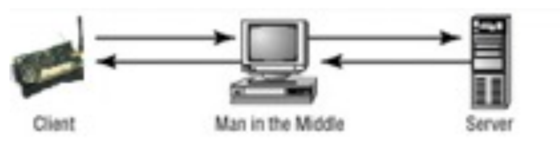
Cloning



Eavesdropping

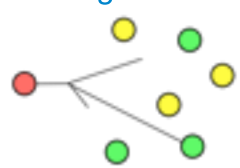


Man-in-the-Middle attack

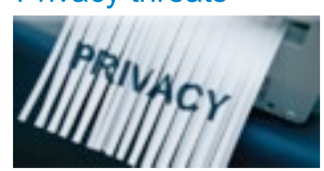


DoS attack

Routing attack



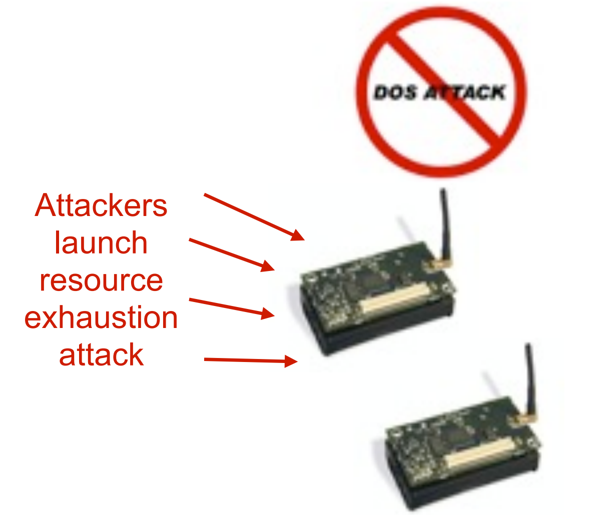
Privacy threats

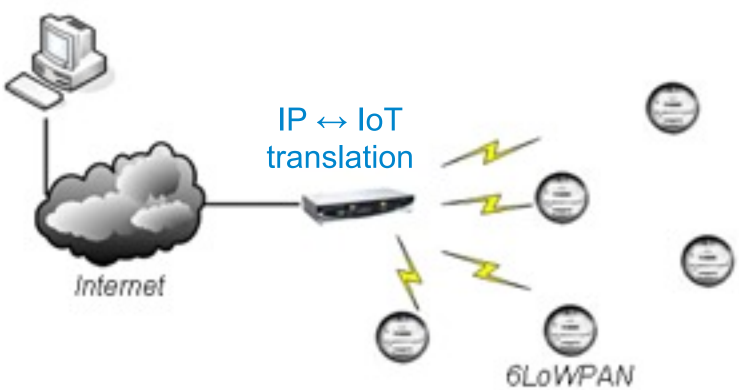


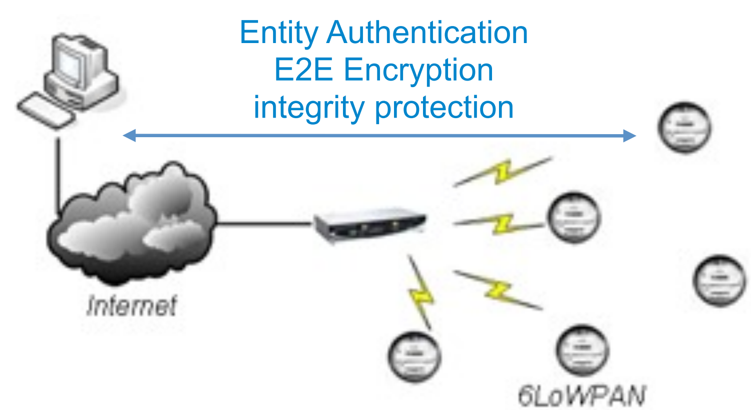
Firmware Replacement

Features and requirements (1/2)

Bootstrapping	Operation
Incremental deployment	End-to-End security
Identity and key management	Mobility support
Privacy-aware identification	Group membership management
Resource-constraints , DoS	

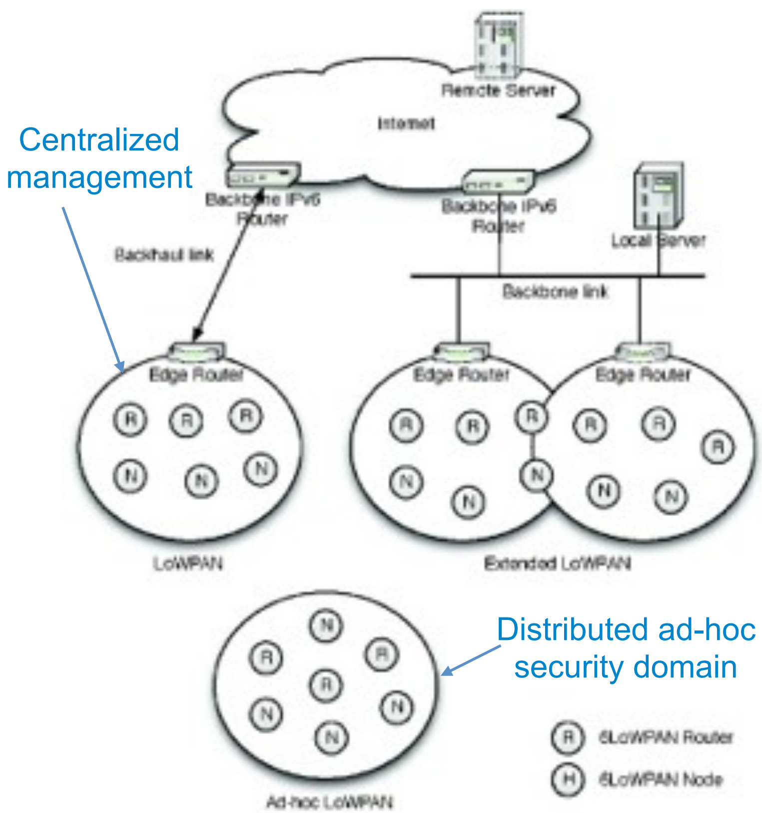




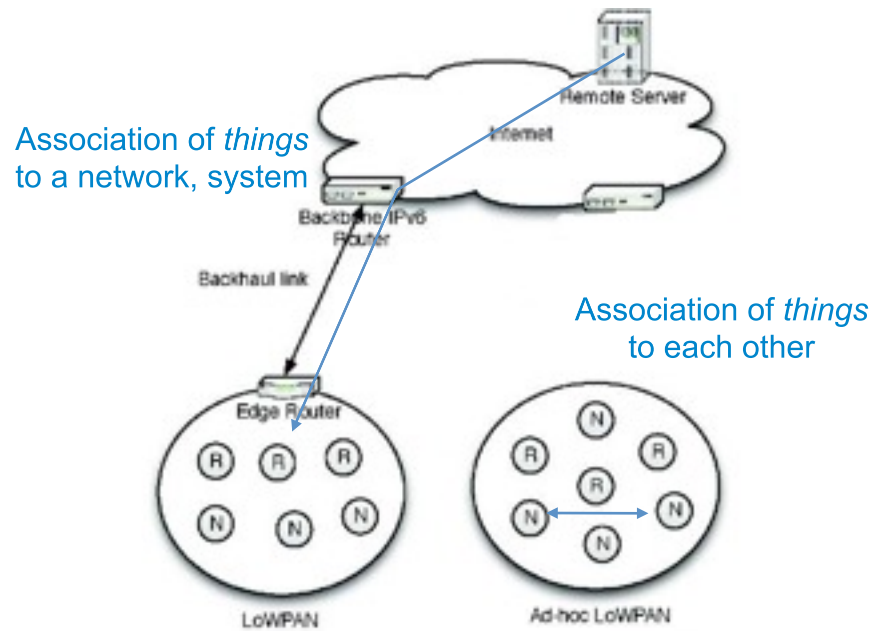


Features and requirements (2/2)

Distributed vs. Centralized architecture



Bootstrapping a thing's identity and keying materials



Discussion

- Definition of a *standard* lightweight bootstrapping protocol
  - rawPublicKey (already defined in CoAP draft)
  - PSK (support for constrained devices, e.g., Class 0 and Class1 [10/100])
- Assessment of security mechanisms (security, operation & performance)
  - DTLS implementation on constrained devices
- Flexible security architecture and operational policies
  - *Roll security framework* and *AMIKEY* (6lowPAN)

Questions

CoAP Security Options

`draft-yegin-coap-security-options-00`

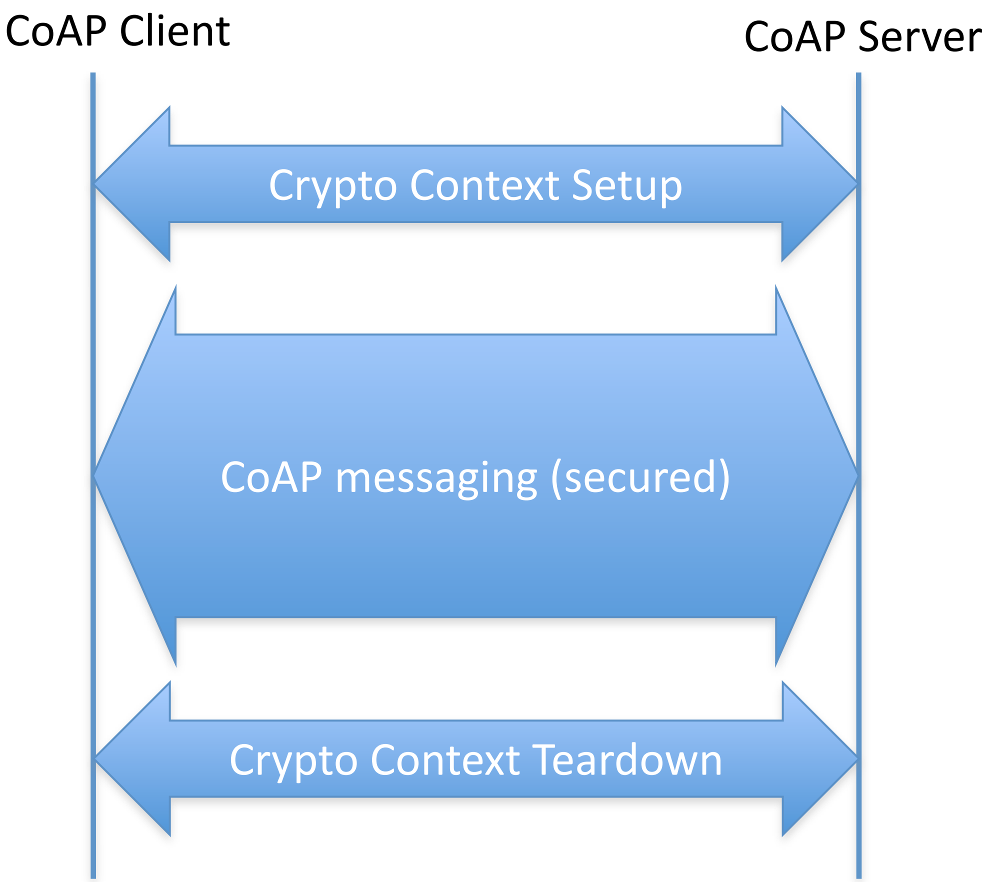
Alper Yegin, Zach Shelby

IETF 82

Objective

- Current solutions: DTLS and IPsec
- Goal: Define a CoAP-layer security solution as an efficient/flexible alternative.
  - A PSK-based solution. Key management (i.e., provisioning of PSK) is a separate issue and not in scope of this I-D.

In a Nutshell

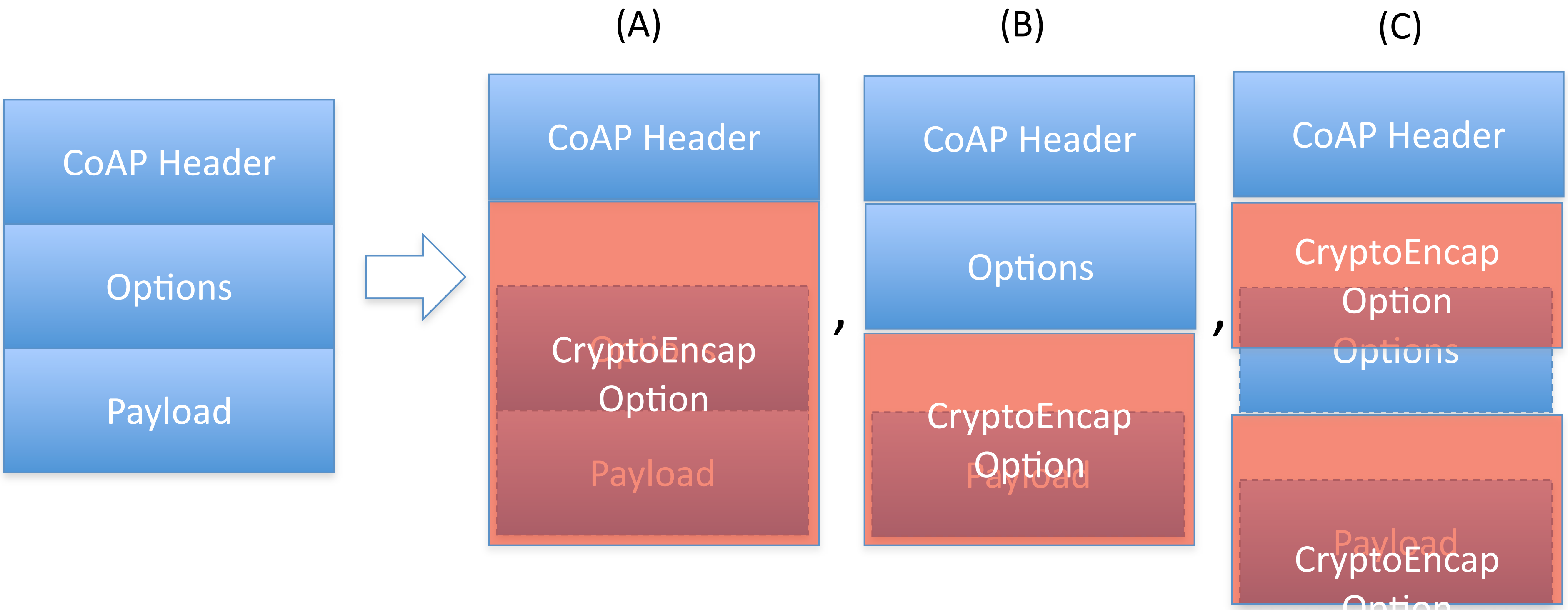


- *CryptoInitiate Option* to setup context
- *CryptoEncap Option* to carry a MAC and envelop other options/ payload in encrypted form.
- *CryptoTerminate Option* to teardown the context.

Crypto Context

- Context ID
- Key Name
- Crypto Algorithm (AES-CCM, AES-CTR, etc.)
- Nonce

Protected Messages



Unprotected message

Whole message integrity protected, options + payload encrypted

Whole message integrity protected, payload encrypted

Using two separate crypto contexts.

Efficiency/Flexibility

- Less per-packet overhead
  - *Excluding crypto-driven overhead (padding, nonces, MAC) that is common…*
  - DTLS: **5 bytes** = 1 (content type) + 2 (version) + 2 (length fields in TLS record)
  - IPsec: **9 bytes** = 4 (SPI) + 4 (seq.no) + 1 (nxt.hdr)
  - CoAP-Sec: **3 or 4 bytes** = 1 or 2 (option hdr) + 1 (context id) + 1 (opt.cnt)
- Less signaling to setup
  - DTLS: **2** round-trips.
  - IKEv2: **2** round-trips.
  - CoAP-Sec: **1** round-trip. (reduce to **0** by piggybacking???)
- Less crypto processing
  - Per-message, per-option/payload selective encryption.
- Less code (by how much???)
  - As a special-purpose integrated solution.
- More flexible
  - Control over individual option/payload treatment.
  - Leaving some options in the clear for intermediary processing.

Known Issues/Improvements

- CryptoInitiate and CryptoTerminate should be integrity protected (requires keys derived from the PSK).

Feedback

82nd IETF: core WG Agenda

15:20	Introduction, Agenda, Status	Chairs (10)
15:30	1 – core CoAP	ZS (90)
17:00	1 – block, observe	CB (20)
17:20 retire to Friday, 11:20	Intro	Chairs (05)
11:25	1 – core CoAP: artificial limitations	CB (10)
11:35	Group Communication	AR (10)
11:43	Naming, Data Formats	JA (10)
11:57	Discovery	ZC MI ZS
12:27	Using CoAP/Naming	KL (8)
12:35	HTTP Mapping	AR (5)
12:40	security	SLK AY BS
13:10	new: CoAP-over-X	KL (10)
13:20	next steps	Chairs (10)
13:30 retire		

CoAP Over SMS

draft-li-core-coap-over-sms-00.txt

Kepeng Li

Mapping to SMS messages

```
MS/CoAP CLIENT        Service Center            MS/CoAP SERVER
      |                       |                         |
      |   ---SMS-SUBMIT--->   |                         |
      | <-SMS-SUBMIT-REPORT-- |                         |
      |                       |                         |
      |                       |   --SMS-DELIVER--->     |
      |                       | <-SMS-DELIVER-REPORT--  |
      |                       |                         |
      | <-SMS-STATUS-REPORT-- |                         |
      |                       |                         |
```

Figure 1: CoAP Messages over SMS

Some Considerations

- ✓ Addressing
  - ✓Use MSISDN
- ✓Options Mapping
  - ✓Use MSISDN as URI-Host
  - ✓URI-Port MUST NOT be included
- ✓Next Step
  - ✓Merge with draft-becker-core-coap-sms-gprs-00

Transport of CoAP over SMS and GPRS
draft-becker-core-coap-sms-gprs-00

Markus Becker, Koojana Kuladinithi, Thomas Pötsch

CoRE WG, IETF-82, Taipei

Scenarios

- M2M communication frequently based on SMS (and GPRS)
- SMS for wakeup from power-save mode

```
            CoAP-REQ
+------+     (SMS)     +------+
|  A   |   ------->    |  B   |
|(cell)|   <-------    |(cell)|
+------+   CoAP-RES    +------+
             (SMS)
```

```
            CoAP-REQ
+------+     (SMS)     +------+
|  A   |   ------->    |  B   |
|(cell)|   <-------    |(cell)|
+------+   CoAP-RES    +------+
            (GPRS)
```

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

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

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

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)
  - 8 bit: no re-encoding necessary, 140 chars/octets, not well-supported in devices
- Larger Payload: SMS concatenation or coap-block?
- Uri-Host and Uri-Port options MUST NOT be present. Or use of RFC5724 SMS URI scheme ('sms:+15105550101')?
- Higher default RESPONSE_TIMEOUT
- No Multicast
- New Options Reply-To-Uri-Host/Port for GPRS return path? (coap-misc User Options?)
- Proxying?

82nd IETF: core WG Agenda

15:20	Introduction, Agenda, Status	Chairs (10)
15:30	1 – core CoAP	ZS (90)
17:00	1 – block, observe	CB (20)
17:20 retire to Friday, 11:20	Intro	Chairs (05)
11:25	1 – core CoAP: artificial limitations	CB (10)
11:35	Group Communication	AR (10)
11:43	Naming, Data Formats	JA (10)
11:57	Discovery	ZC MI ZS
12:27	Using CoAP/Naming	KL (8)
12:35	HTTP Mapping	AR (5)
12:40	security	SLK AY BS
13:10	new: CoAP-over-X	KL (10)
13:20	next steps	Chairs (10)
13:30 retire		

82nd IETF: core WG Agenda

15:20	Introduction, Agenda, Status	Chairs (10)
15:30	1 – core CoAP	ZS (90)
17:00	1 – block, observe	CB (20)
17:20 retire to Friday, 11:20	Intro	Chairs (05)
11:25	1 – core CoAP: artificial limitations	CB (10)
11:35	Group Communication	AR (10)
11:43	Naming, Data Formats	JA (10)
11:57	Discovery	ZC MI ZS
12:27	Using CoAP/Naming	KL (8)
12:35	HTTP Mapping	AR (5)
12:40	security	SLK AY BS
13:10	new: CoAP-over-X	KL (10)
13:20	next steps	Chairs (10)
13:30 retire		