Constrained RESTful Environments WG (core)

Chairs:

Cullen Jennings <fluffy@cisco.com>

Carsten Bormann <cabo@tzi.org>

With some help from Peter Saint-Andre...

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)

IAB Workshop and Tutorial on Interconnecting Smart Objects with the Internet (2011-03-25/-26)

- http://www.iab.org/about/workshops/smartobjects/tutorial.html
- http://www.iab.org/about/workshops/smartobjects/index.html

- See position papers and slides there
- IAB workshop report to come out

- One thing became clear that may be worth reporting about:

10/100 vs. 50/250

- There is not just a single class of “constrained node”
- Class 0: too small to securely run on the Internet
  - “too constrained”
- Class 1: ~10 KiB data, ~100 KiB code
  - “quite constrained”, “10/100”
- Class 2: ~50 KiB data, ~250 KiB code
  - “not so constrained”, “50/250”
- These classes are not clear-cut, but may structure the discussion and help avoid talking at cross-purposes

Milestones (from WG charter page)

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

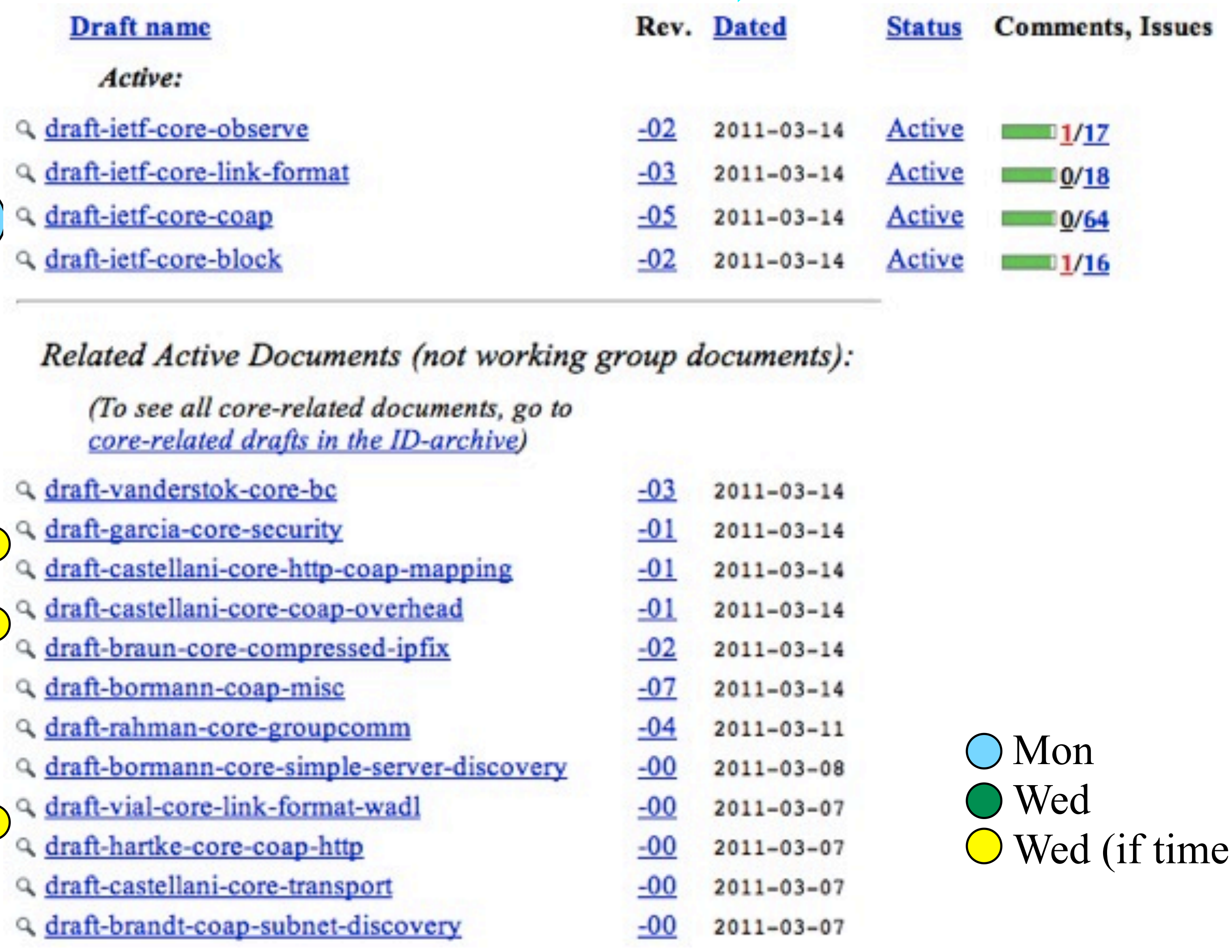
Document submissions to IESG:

- **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/



	Draft name	Rev.	Dated	Status	Comments, Issues
	Active:				
🔵	draft-ietf-core-observe	-02	2011-03-14	Active	🟩 1/17
🔵	draft-ietf-core-link-format	-03	2011-03-14	Active	🟩 0/18
🔵	draft-ietf-core-coap	-05	2011-03-14	Active	🟩 0/64
🔵	draft-ietf-core-block	-02	2011-03-14	Active	🟩 1/16

Related Active Documents (not working group documents):

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

	Draft name	Rev.	Dated
🟢	draft-vanderstok-core-bc	-03	2011-03-14
🔵🟡	draft-garcia-core-security	-01	2011-03-14
🟢	draft-castellani-core-http-coap-mapping	-01	2011-03-14
🟡	draft-castellani-core-coap-overhead	-01	2011-03-14
	draft-braun-core-compressed-ipfix	-02	2011-03-14
	draft-bormann-coap-misc	-07	2011-03-14
🟢	draft-rahman-core-groupcomm	-04	2011-03-11
🔵	draft-bormann-core-simple-server-discovery	-00	2011-03-08
🟡	draft-vial-core-link-format-wadl	-00	2011-03-07
🟢	draft-hartke-core-coap-http	-00	2011-03-07
	draft-castellani-core-transport	-00	2011-03-07
🔵	draft-brandt-coap-subnet-discovery	-00	2011-03-07
🟡	draft-moritz-core-soap-over-coap-00		

🔵 Mon
🟢 Wed
🟡 Wed (if time permits)

80th IETF: core WG Agenda

13:00	Introduction, Agenda, Status	Chairs (10)
13:10	1 – link-format	ZS (5)
13:15	1 – core CoAP	ZS (25)
13:40	1 – block	CB (15)
13:55	1 – observe	KH (25)
14:20	2 – basic security	ZS (20)
14:40	discovery	CB+AB (20)
15:00 retire to Wednesday, 15:10	Intro	Chairs (05)
15:15	Group Communication	AR (10)
15:25	CoAP Usage	PV (10)
15:35	HTTP Mapping	KH+AC (15)
15:50	new: SOAP; WADL; overhead; security	GM ZS AC OG (5 each)
16:10 retire		

80th IETF: core WG Agenda

13:00	Introduction, Agenda, Status	Chairs (10)
13:10	1 – link-format	ZS (5)
13:15	1 – core CoAP	ZS (25)
13:40	1 – block	CB (15)
13:55	1 – observe	KH (25)
14:20	2 – basic security	ZS (20)
14:40	discovery	CB+AB (20)
15:00 retire to Wednesday, 15:10	Intro	Chairs (05)
15:15	Group Communication	AR (10)
15:25	CoAP Usage	PV (10)
15:35	HTTP Mapping	KH+AC (15)
15:50	new: SOAP; WADL; overhead; security	GM ZS AC OG
16:10 retire		(5 each)

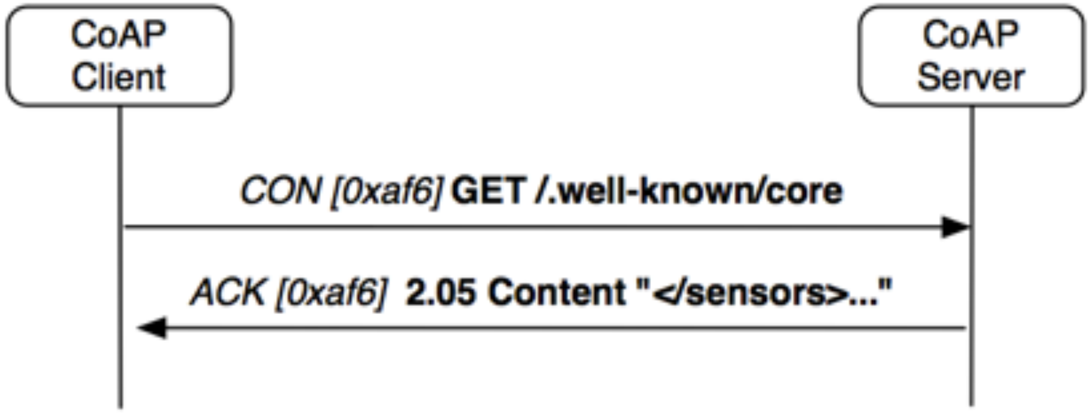
CoRE Link Format

draft-ietf-core-link-format-03

Z. Shelby

CoRE WG, IETF-80 Prague

What is the CoRE Link Format


CoAP Client
CoAP Server
CON [0xaf6] GET /.well-known/core
ACK [0xaf6] 2.05 Content "</sensors>..."

```
REQ: GET /.well-known/core

RES: 2.05 Content
</sensors>;ct=40;rt="index";rt="Sensor Index",
</sensors/temp>;rt="TemperatureC";if="sensor",
</sensors/light>;ct=41;rt="LightLux";if="sensor",
<http://www.example.com/sensors/t123>;anchor="/sensors/temp"
;rel="describedby",
</t>;anchor="/sensors/temp";rel="alternate"
```

Progress since Beijing

- WGLC completed on Jan 26th
- Thanks everyone for the comments
- 9 tickets identified during LC
- Editorial improvements
  - Text about alternative web linking formats
  - Added use case examples
    - Discovery
    - Resource Collections
    - Resource Directory
  - Clarified difference to RFC5988

Using WADL in Interface Descriptions

```
REQ: GET /.well-known/core

RES: 2.00 OK

</tmp>;n="AirTemperature"; d="http://www.example.org/ha1.wadl#sensor.temperature",

</tmp/thr>;n="TemperatureAlarm"; d="http://www.example.org/
    ha1.wadl#parameter.threshold"

</pwr>;n="PowerConsumption"; d="http://www.example.org/em2.wadl#meter.power",

</pwr/thr>;n="PowerAlarm"; d="http://www.example.org/em2.wadl#parameter.threshold"
```

```
<application xmlns="http://wadl.dev.java.net/2009/02" xmlns:xsi="http://www.w3.org/
    2001/XMLSchema-instance" xsi:schemaLocation="http://wadl.dev.java.net/2009/02
    wadl.xsd"> <resource_type id="meter.power">

          <method name="GET" id="GetPower">

                    <request> <representation mediaType="application/xml" /> </request>
          </method> </resource_type>

    <resource_type id="parameter.threshold">

          <method name="PUT" id="SetThreshold> <request> <representation
    mediaType="application/xml" /> </request>

          </method> </resource_type>
```

See draft-vial-core-link-format-wadl-00

Current Status

- RFC5988 “Web Linking” published
  - The CoRE link format is derived from this
- Document has been stable for 6 months
- Tested in two plugfest events
  - Only trivial issues found in Beijing, fixed in -03
  - Has been universally implemented
- No known open issues

80th IETF: core WG Agenda

13:00	Introduction, Agenda, Status	Chairs (10)
13:10	1 – link-format	ZS (5)
13:15	1 – core CoAP	ZS (25)
13:40	1 – block	CB (15)
13:55	1 – observe	KH (25)
14:20	2 – basic security	ZS (20)
14:40	discovery	CB+AB (20)
15:00 retire to Wednesday, 15:10	Intro	Chairs (05)
15:15	Group Communication	AR (10)
15:25	CoAP Usage	PV (10)
15:35	HTTP Mapping	KH+AC (15)
15:50	new: SOAP; WADL; overhead; security	GM ZS AC OG
16:10 retire		(5 each)

Constrained Application Protocol

draft-ietf-core-coap-05

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

CoRE WG, IETF-80 Prague

Progress since Beijing

- Plugfest held in Beijing on coap-04
  - Tested with –observe and –block
- -05 released based on Beijing WG feedback
  - 13 tickets closed plus editorial work
- So what did we change?
  - Document was reorganized for easier reading
  - Several editorial improvements
  - Minor technical changes
  - IANA scheme registration improved

Editorial Improvements

- Piggy-backed response term added
  - No more deferred responses
- Section added on message size implementation considerations
- Clarification made on human readable error payloads
- Definition of CoAP methods improved
- Clarified the uniqueness of tokens
- Clarified relation between elective/critical and option numbers

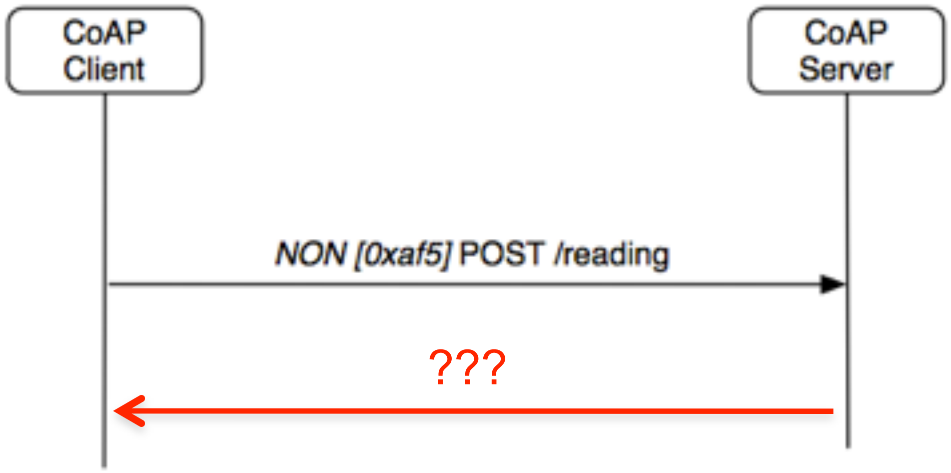
Technical Improvements

- GET requests now return a 2.05 (Content) response instead of 2.00 (OK) response (#104)
- Added text to allow 2.02 (Deleted) responses in reply to POST requests (#105)
- Improved message deduplication rules (#106)
- Max-Age removed from requests (#107)
- Location-Query Option added (#113)
- ETag length set to 1-8 bytes (#123)

Current Issues

- How to respond to a NON request?
- UTF8 URI matching needs clarification
- HTTP mapping section
  - Separating standards part from implementation advice
- Security updates

How to respond to a NON request?



- Right, that doesn't work…
- coap-05 current ignores this
- **Ticket:** Clarify that a NON is not ACKed, however a response still is sent (on a new message exchange).

UTF8 URI matching needs clarification

- We specify the use of UTF-8 compatible URIs in Net-Unicode Form [RFC5198]
  - Why? HTTP mapping compatibility
- Comparing arbitrary UTF-8s in a constrained device would not be sane
- The intention is that CoAP implementations simply do an opaque binary comparison of URIs
- **Ticket:** Section 6.2 should make that more explicit

HTTP mapping section

- Section 8 defines a CoAP-HTTP mapping
- The current description goes into more detail than is needed in the protocol specification
- draft-hartke-core-coap-http-00 suggests
  - Simple standard text (Section 2)
  - Implementation considerations and examples
- **Ticket:** Adopt the simple standard text for core-coap, create a new HTTP mapping guide I-D with implementation considerations and examples with the WG

Security Updates

- Currently unclear if coap-05 is specifying IPSec or DTLS, and in what modes
- **Ticket:** Clarify that DTLS is the must-implement security mechanism for CoAP
  - SharedKey (PSK), MultiKey (PSK) and Certificate mode are related to DTLS
  - Pre-shared key mode is must implement
  - NoSec mode (thus no DTLS) may be used in combination with IPSec, L2 encryption or cases where security really isn't needed
  - IPSec section still needed, but to be moved and edited appropriately.

Security Updates

- **Ticket:** Must implement cipher suites need updates and alignment with ZigBee IP
  - Remove SHOULD wording before each…
  - TLS_PSK_WITH_AES_128_CCM_8 must implement for SharedKey and MultiKey modes
  - TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 must implement for Certificate mode
- **Ticket:** IPSec section should reference draft-kivinen-ipsecme-ikev2-minimal (informational)
- **Ticket:** Clarify the interaction of DTLS authentication and resource access control

Next Steps

- Document has been stable since Beijing
  - Mainly editorial improvements in -05
- Tested in two plugfest events
- Well implemented (20+ implementations?)
- A few minor updates identified

80th IETF: core WG Agenda

13:00	Introduction, Agenda, Status	Chairs (10)
13:10	1 – link-format	ZS (5)
13:15	1 – core CoAP	ZS (25)
13:40	1 – block	CB (15)
13:55	1 – observe	KH (25)
14:20	2 – basic security	ZS (20)
14:40	discovery	CB+AB (20)
15:00 retire to Wednesday, 15:10	Intro	Chairs (05)
15:15	Group Communication	AR (10)
15:25	CoAP Usage	PV (10)
15:35	HTTP Mapping	KH+AC (15)
15:50	new: SOAP; WADL; overhead; security	GM ZS AC OG (5 each)
16:10 retire		

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

2048

- Just an accident
- Not compatible with CoAP MTU of 1152

- Relegate code point szx = 7 to Reserved

The block option vs. methods

- **GET: trivial**
  - **Receiver: watch Etag to obtain parts of same resource repr.**
  - **Also works for asynchronous responses (subscriptions)**
    - initiative is with responder, then!
- **PUT, POST: trigger actual update on M=0**
  - **manage parallel operations based on token option**
- **Block is CRITICAL**

#45 large responses to POST/PUT

- **Block can be used either on request body (POST/PUT) or on response body (GET), not both**
  - **do we need large POST/PUT responses? If yes:**
  - **add a second option?**

Wait a minute...

- **We don't *have* response bodies to PUT or POST**
- **PUT/POST: 2.04 (Changed), 2.01 (Created)**
- **"**Methods with payload are PUT and POST**, and the response codes with payload are 2.05 (Content)** and the error codes."

- **Enable payload with 2.04, 2.01**
- **Provide generic 2.00 OK for generic POST responses**

Block option

- "In the response for a GET or in the request for a PUT or POST, the Block option describes..."
  - Method code influences semantics of option
  - Could add another option for PUT/POST response... Naah.
- Clean this up:
- Block1 option for request payloads (PUT, POST)
- Block2 for response payloads (GET, PUT, POST)
- (Allocate new numbers for both to provide some grace period in testing; free old number 13 on July 1st.)

80th IETF: core WG Agenda

13:00	Introduction, Agenda, Status	Chairs (10)
13:10	1 – link-format	ZS (5)
13:15	1 – core CoAP	ZS (25)
13:40	1 – block	CB (15)
13:55	1 – observe	KH (25)
14:20	2 – basic security	ZS (20)
14:40	discovery	CB+AB (20)
15:00 retire to Wednesday, 15:10	Intro	Chairs (05)
15:15	Group Communication	AR (10)
15:25	CoAP Usage	PV (10)
15:35	HTTP Mapping	KH+AC (15)
15:50	new: SOAP; WADL; overhead; security	GM ZS AC OG (5 each)
16:10 retire		

observe: status

- **Basic concepts are stable**
- **Discussion about the server-side state needed**
  - **per observation relationship:**
    - IP address + port number
    - Token
    - Lifetime (timer, running down)

Removing the lifetime

- We never had a really good use case
- Observation relationship already terminates when:
  - CON response answered by RST
  - node not reachable
  - error on server side (final notification)
  - re-install/remove observation relationship

- Change “lifetime” to “observe”
- An Observation relationship is now only terminated in the above cases

Without lifetime, how to detect reordering?

- Detection needed in non-continual cases to enable eventual consistency
- Was using lifetime counters
- Now using a single per-server counter
  - roughly counting seconds
  - sequence number arithmetic
  - no potential for reordering between times sequence number wraps...
  -

Keepalives?

- Well-known problem
  - garbage-collect at server side
  - actually get the notifications on client side
- Taking away lifetime takes away source of expectation for timescales
- Do we try to set such an expectation?
  - 24 hours?

80th IETF: core WG Agenda

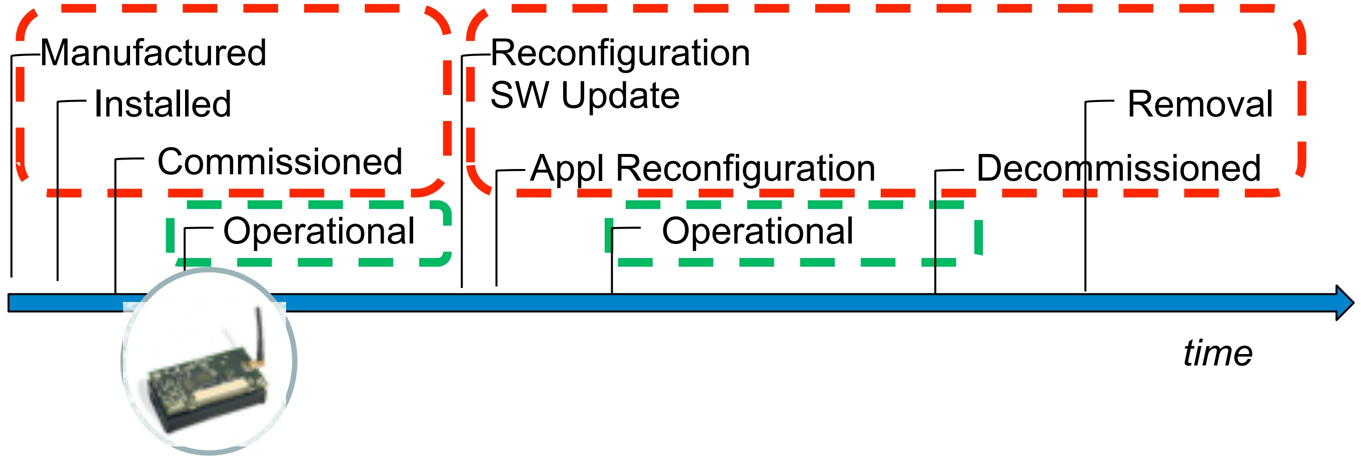
13:00	Introduction, Agenda, Status	Chairs (10)
13:10	1 – link-format	ZS (5)
13:15	1 – core CoAP	ZS (25)
13:40	1 – block	CB (15)
13:55	1 – observe	KH (25)
14:20	2 – basic security	ZS (20)
14:40	discovery	CB+AB (20)
15:00 retire to Wednesday, 15:10	Intro	Chairs (05)
15:15	Group Communication	AR (10)
15:25	CoAP Usage	PV (10)
15:35	HTTP Mapping	KH+AC (15)
15:50	new: SOAP; WADL; overhead; security	GM ZS AC OG (5 each)
16:10 retire		

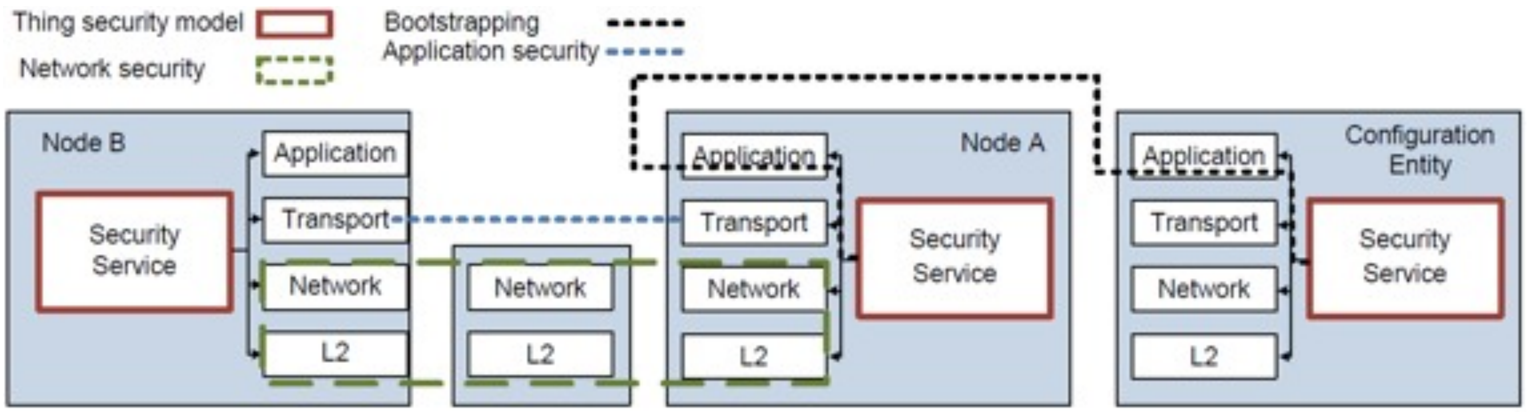
CoAP Security

Zach Shelby, Oscar Garcia-Morchon, Cullen Jennings, Carsten Bormann

CoRE WG, IETF-80 Prague

Thing lifecycle and security framework





See draft-garcia-core-security

CoAP Security

- draft-ietf-core-coap must satisfy BCP 61
- How are we doing on that?
  - ✓ Security mechanism: DTLS
  - ✓ DTLS mode: PSK (must-implement), Certificates
  - ✓ Must-implement cipher suite for each
- What about key management in PSK mode?
  - draft-sarikaya-core-sbootstrapping is a nice survey of techniques
  - draft-garcia-core-security is great background
- Now we really need one must-implement solution

Security Bootstrapping Requirements

- Must work on “10/100” devices
- Simple constrained networks
- Support for DTLS PSK mode of CoAP
  - SharedKey and MultiKey modes
- Pre-installing, imprinting and updating keys
- Provide interoperability
- Each device needs:
  - One or more keys for use with DTLS
  - psk_identity per key (the correct key to use)

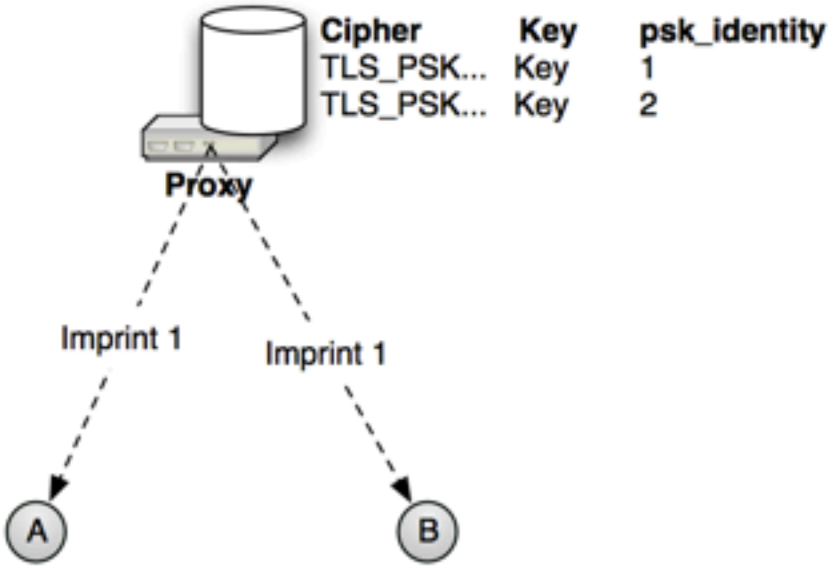
What are our options?

- Basic Security Bootstrapping
  - Specify something simple in CoRE
  - Simple mother-duckling
    - Don’t specify how a node gets into duckling mode
    - Could be battery insert, button press etc.
  - Pre-installed key or imprint/update over CoAP
- PANA/EAP-TTLSv0 [RFC5281]
  - Network access control
  - TTLS could be used to distribute keys
  - Used by ZigBee IP
  - Overkill as must-implement?

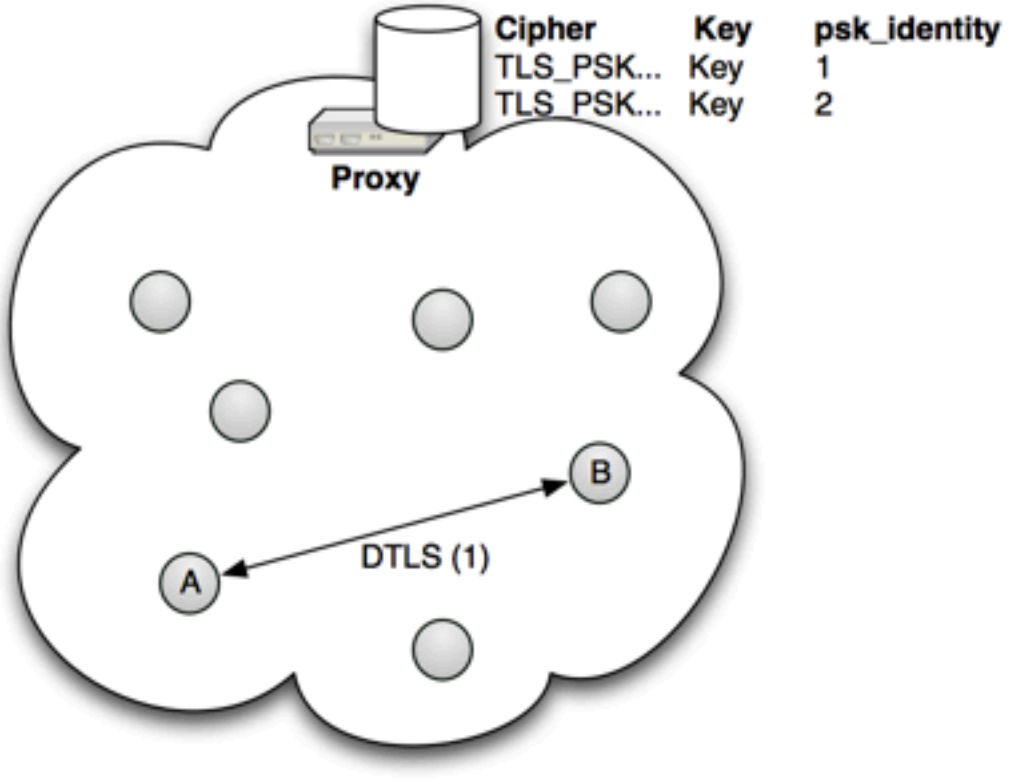
Basic Security Bootstrapping (strawman)

- Storing key+ID as CoAP resources
  - /.well-known/core-keys
- Mother-duckling (mutual!)
- Local proxy as a security manager
- Discovering new devices
  - Network layer information (e.g. ND, routing)
  - CoRE discovery
- Installing/updating keys
  - PUT /.well-known/core-keys/{identity}/TLS_PSK...
- Default root identity for authentication

Basic Security Bootstrapping



Cipher	Key	psk_identity
TLS_PSK...	Key	1
TLS_PSK...	Key	2



Cipher	Key	psk_identity
TLS_PSK...	Key	1
TLS_PSK...	Key	2

Conclusions

- Basic security bootstrapping needed
  - We have a charter item to do this
- One focused goal to support CoAP
  - Key mechanism for DTLS PSK
- What is in-scope?
  - A very simple bootstrapping mechanism
- What is out-of-scope?
  - Non-default authorization
  - User-interfaces
  - Everything else

80th IETF: core WG Agenda

13:00	Introduction, Agenda, Status	Chairs (10)
13:10	1 – link-format	ZS (5)
13:15	1 – core CoAP	ZS (25)
13:40	1 – block	CB (15)
13:55	1 – observe	KH (25)
14:20	2 – basic security	ZS (20)
14:40	discovery	CB+AB (20)
15:00 retire to Wednesday, 15:10	Intro	Chairs (05)
15:15	Group Communication	AR (10)
15:25	CoAP Usage	PV (10)
15:35	HTTP Mapping	KH+AC (15)
15:50	new: SOAP; WADL; overhead; security	GM ZS AC OG (5 each)
16:10 retire		

draft-bormann-core-simple-server-discovery

- draft-core-link-format provides resource discovery once a server is known
- how to discover a new server?

- draft-bormann-core-simple-server-discovery describes a server discovery protocol in use in the SAHARA project
  - Thanks: Bengt Kohrt, Julian Kornberger, Henning Müller, Christian Thedieck, Philip Nguyen

Discovery Protocol

- CoAP Server Discovery Server (CSDS): a server that
  - collects resource discovery information, and
  - presents it in integrated form (e.g., in own /.well-known/core)
- New CoAP servers that want to provide discoverable services can make themselves known at the CSDSs
- PUT to coap://CSDS/.well-known/core
  - with link-format as payload
  - empty:
    CSDS is encouraged to GET the client's /.well-known/core

How to find a CSDS?

- Candidate CSDS: An IP address that might or might not be useful for conversion to a CSDS URI

- Find Candidates from:
  - specific static configuration (e.g., anycast addresses), if any,
  - the ABRO option of 6LoWPAN-ND [I-D.ietf-6lowpan-nd],
  - other ND options that might point to servers (e.g., RDNSS),
  - DHCPv6 options that might be defined later.

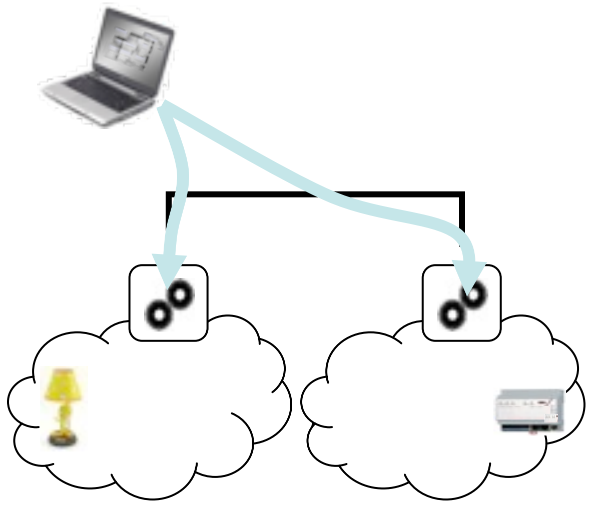
Discovery across subnets

`draft-brandt-coap-subnet-discovery`

Anders Brandt
Sigma Designs
abr@sdesigns.dk

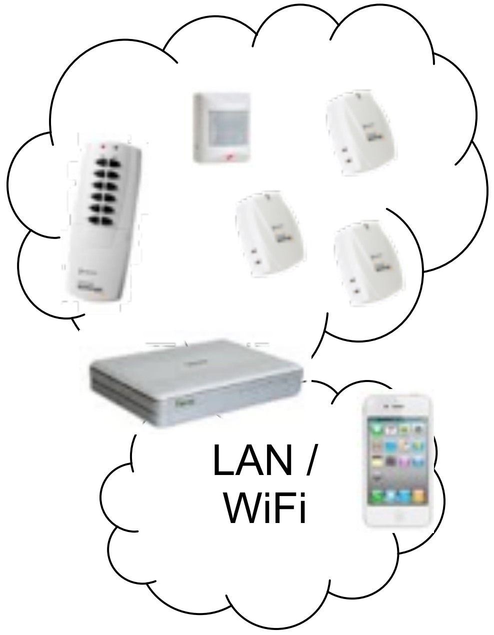
Discovery across subnets

`draft-brandt-coap-subnet-discovery`



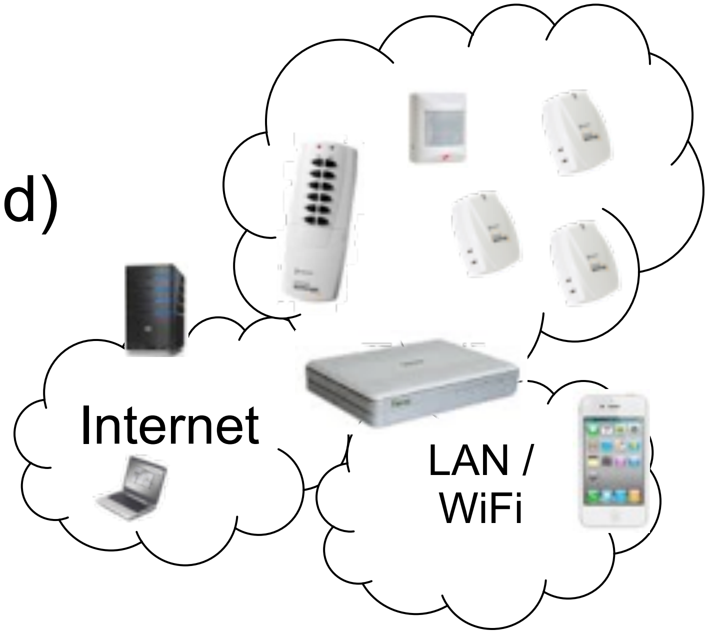
Anders Brandt
Sigma Designs
abr@sdesigns.dk

Scenario: Service Provider bundle



Scenario: Service Provider bundle

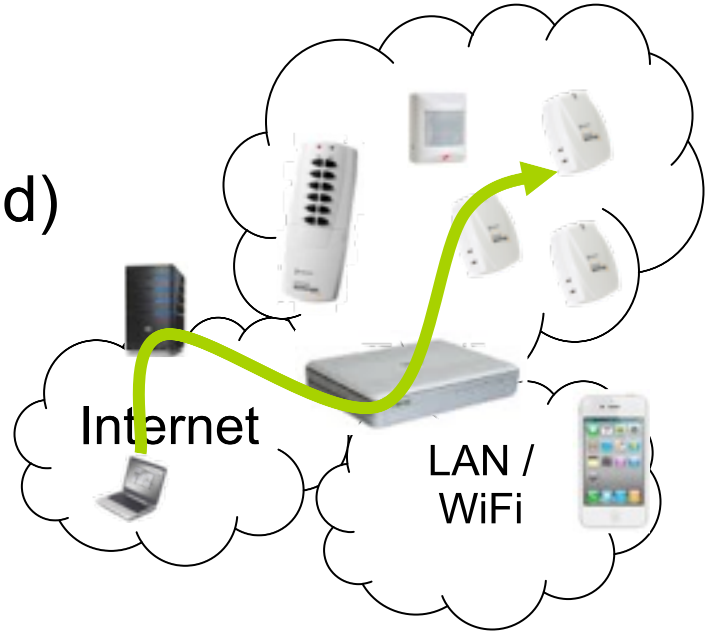
- Border router assigns
  - L2 short addresses
  - Routable IP prefix (autoconfigured)
  - 6LoWPan CIDs



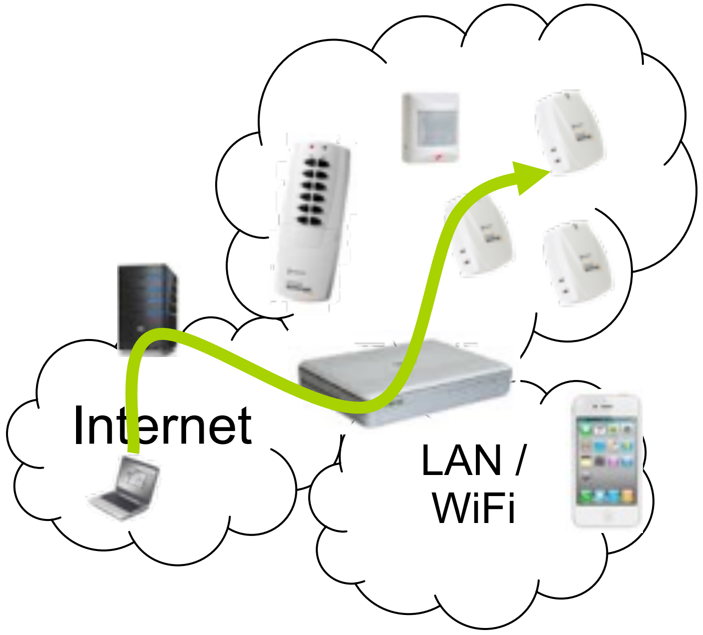
Scenario: Service Provider bundle

- Border router assigns
  - L2 short addresses
  - Routable IP prefix (autoconfigured)
  - 6LoWPan CIDs

- User control
  - Smart phone widgets
  - Web portal
    - Provider: Remote support call center
    - User: Floor plan for drag-n-drop management



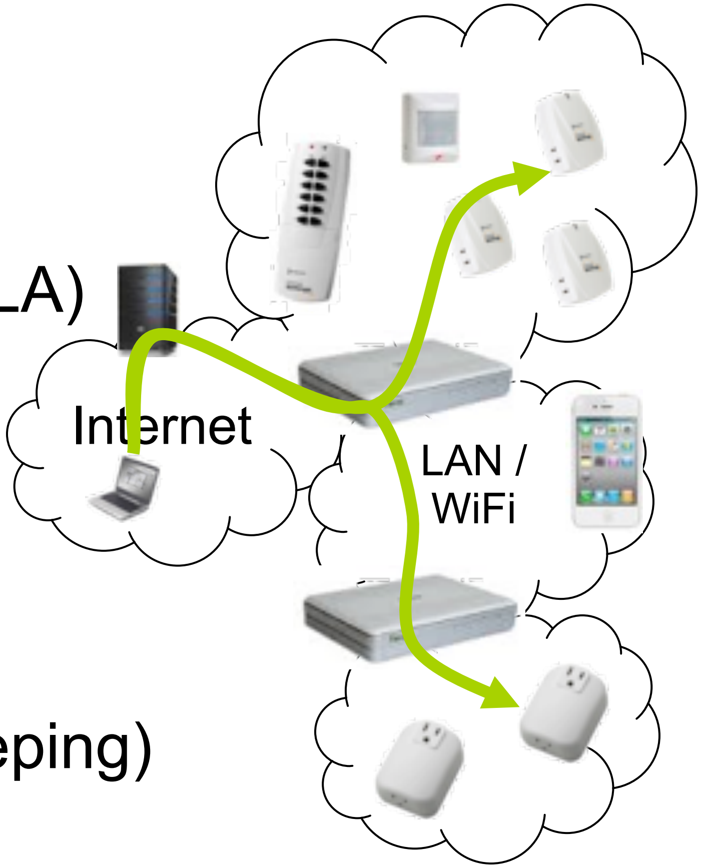
Scenario: Service Provider bundle



Scenario: Service Provider bundle

- Second router added
  - NO user configuration
  - Backbone routing protocol
  - Automatic subnet assignment (ULA)

- Discovery
  - In ALL subnets
  - From any subnet
  - Also battery powered nodes (sleeping)



Requirements

Requirements

- Subnet considerations
  - Discovery client may be anywhere
  - Discovery requests may traverse several subnets
  - Zero-config LLN subnet prefix (ULA)

Requirements

- Subnet considerations
  - Discovery client may be anywhere
  - Discovery requests may traverse several subnets
  - Zero-config LLN subnet prefix (ULA)
- Traffic patterns
  - Avoid LLN multicast
  - Support sleeping nodes
  - Limit LLN traffic

Requirements

Requirements

- Discovery principles
  - Use existing CoAP Link Format principles
  - Support filtered requests
    - ?n=”LightDimmer”

Requirements

- Discovery principles
  - Use existing CoAP Link Format principles
  - Support filtered requests
    - ?n=”LightDimmer”
  - Scalable
    - No data aggregation / synchronization in a single gateway
    - Query more discovery gateways in bigger networks

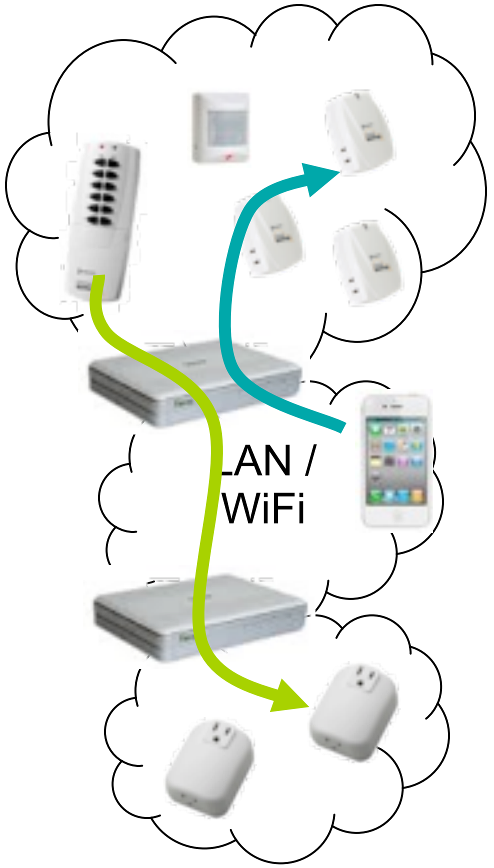
Requirements

- Discovery principles
  - Use existing CoAP Link Format principles
  - Support filtered requests
    - ?n=”LightDimmer”
  - Scalable
    - No data aggregation / synchronization in a single gateway
    - Query more discovery gateways in bigger networks
  - Report legacy devices
    - Protocol must allow discovery of legacy technologies

Requirements

Requirements

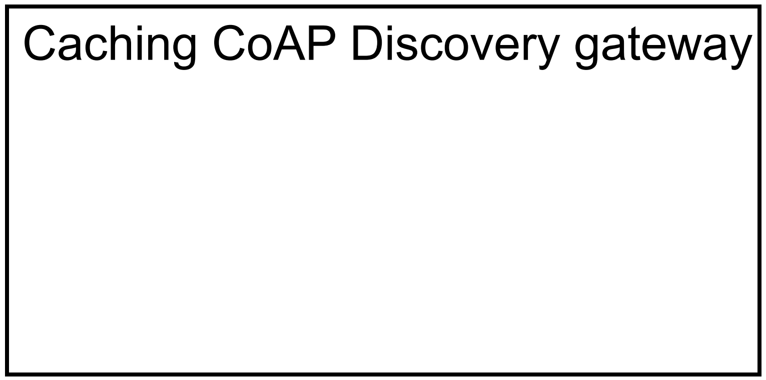
- Zero-configuration
  - Consumer-grade systems ⇒
    - Subnet discovery right out of the box
    - Subnet prefixes must be routable
    - Backbone routing protocol needed



.

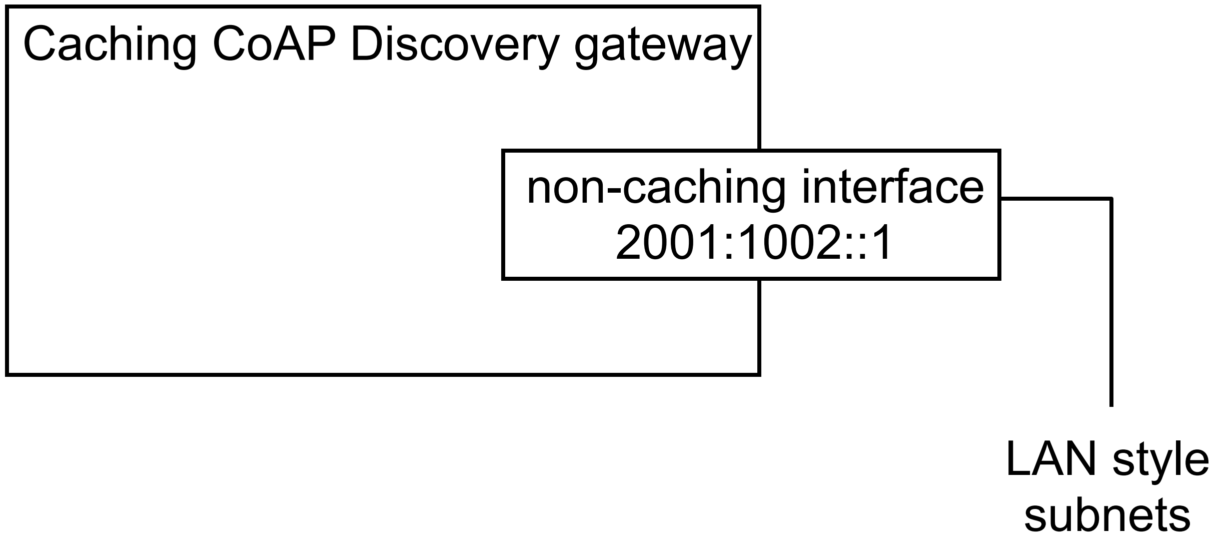
CoAP Discovery

- Two interface types



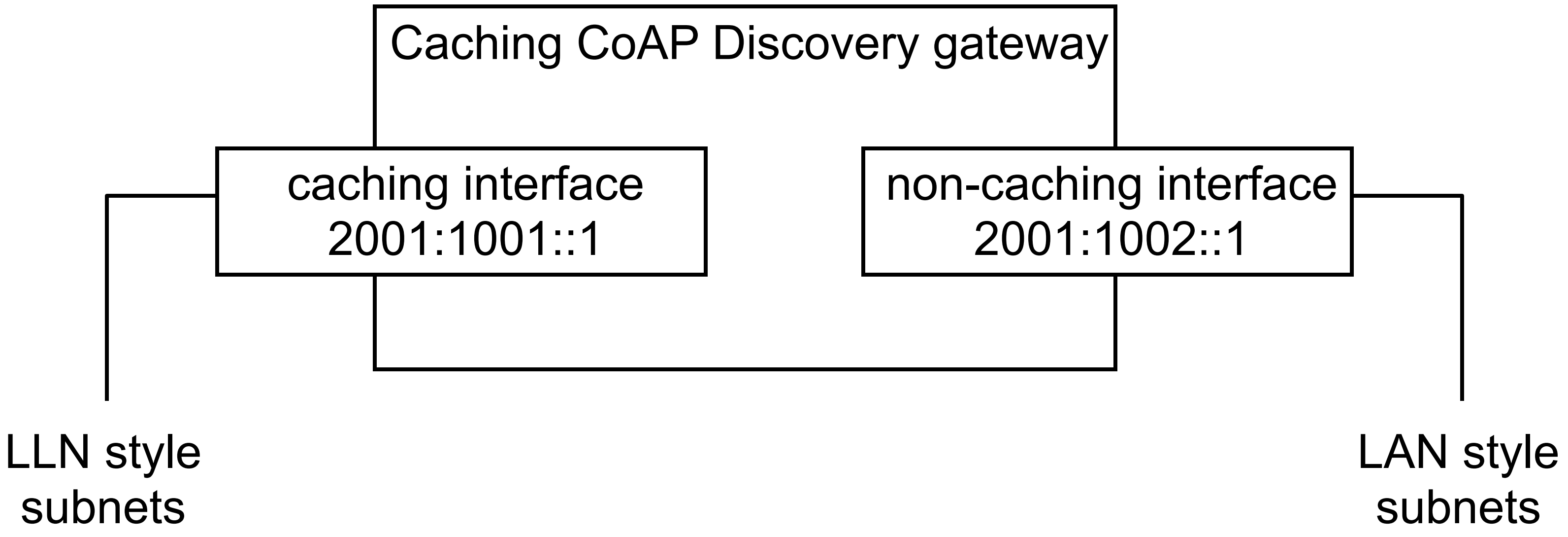
CoAP Discovery

- Two interface types



CoAP Discovery

- Two interface types



CoAP Discovery

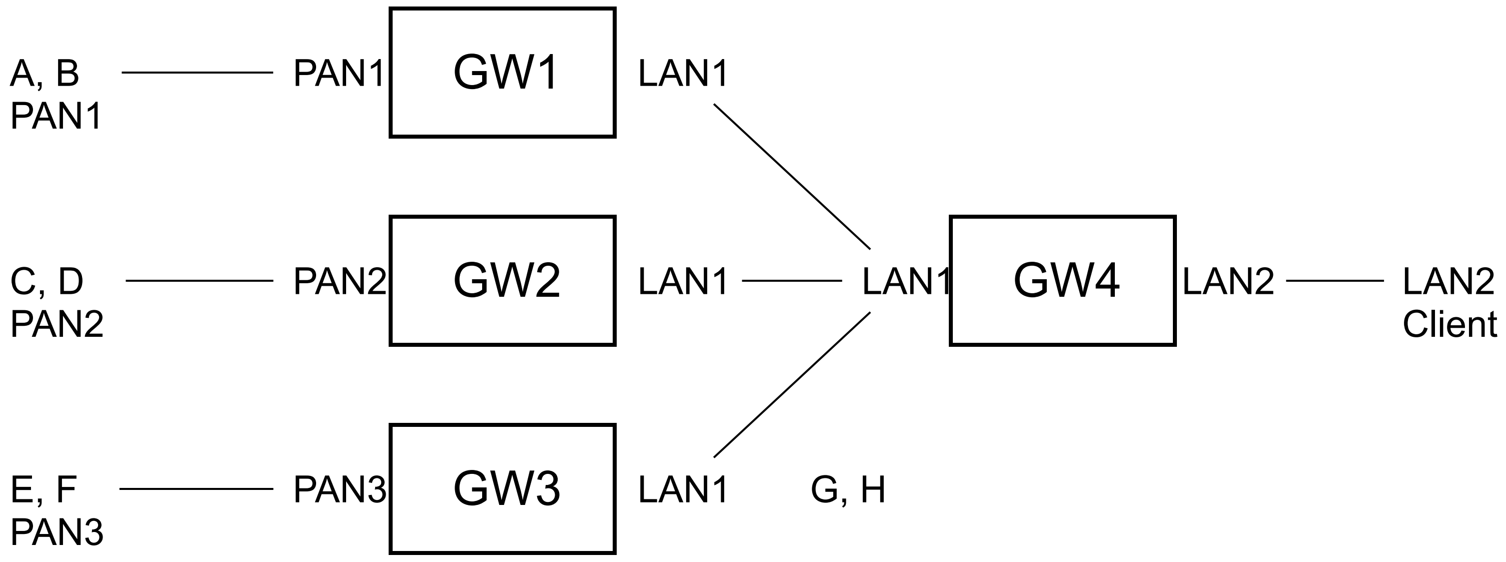
CoAP Discovery

- (non-caching) Discovery Gateway interface
  - Announces an access point to CoAP resources
  - Offers multicast discovery from other subnets
    - E.g. discovery client in LLN, CoAP server in LAN

CoAP Discovery

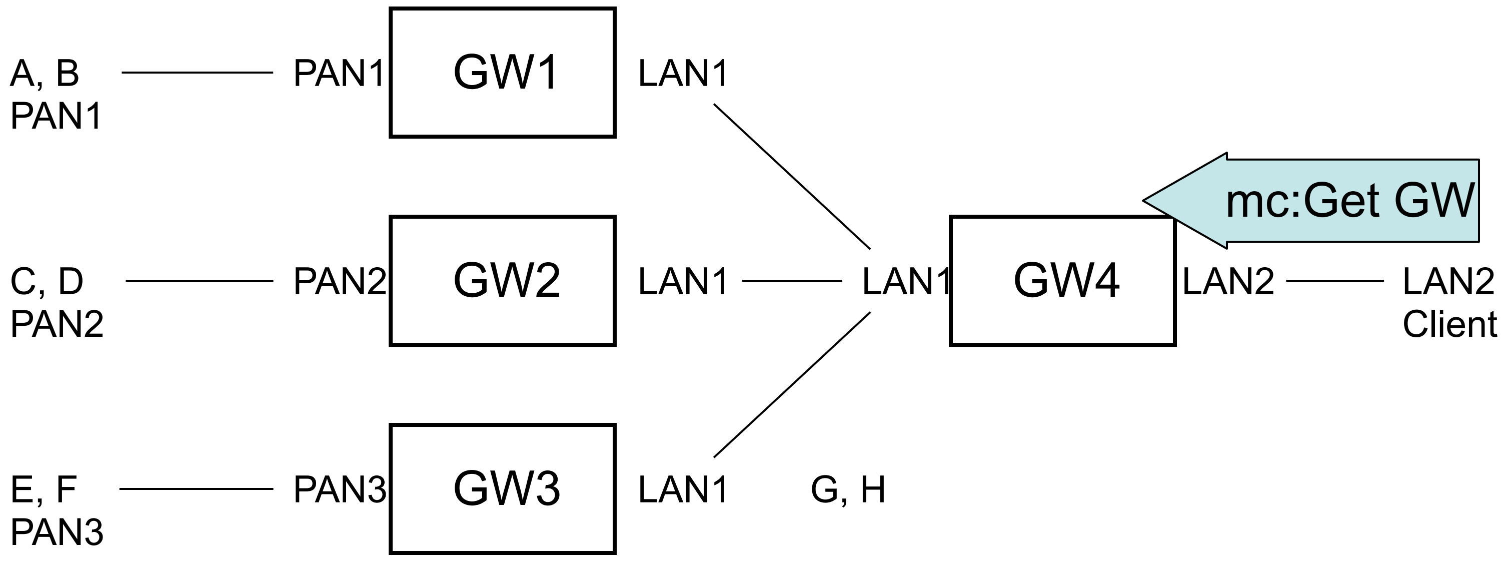
- (non-caching) Discovery Gateway interface
  - Announces an access point to CoAP resources
  - Offers multicast discovery from other subnets
    - E.g. discovery client in LLN, CoAP server in LAN
- **Caching** Discovery Gateway interface
  - Protects LLN bandwidth
    - E.g. discovery client in LAN, CoAP server in multi-hop LLN
  - Eliminates need for multicast
  - Supports sleeping nodes
  - Caches ONLY discovery data

Example: Gateway Discovery



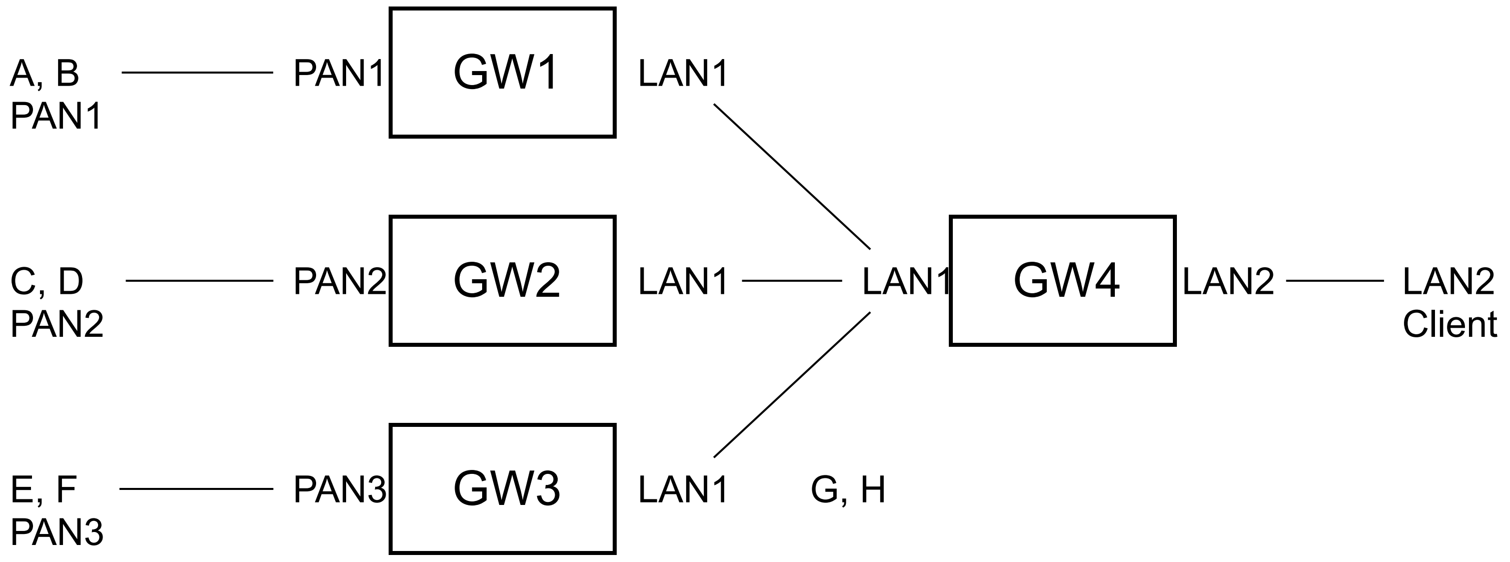
• (on-link)

Example: Gateway Discovery



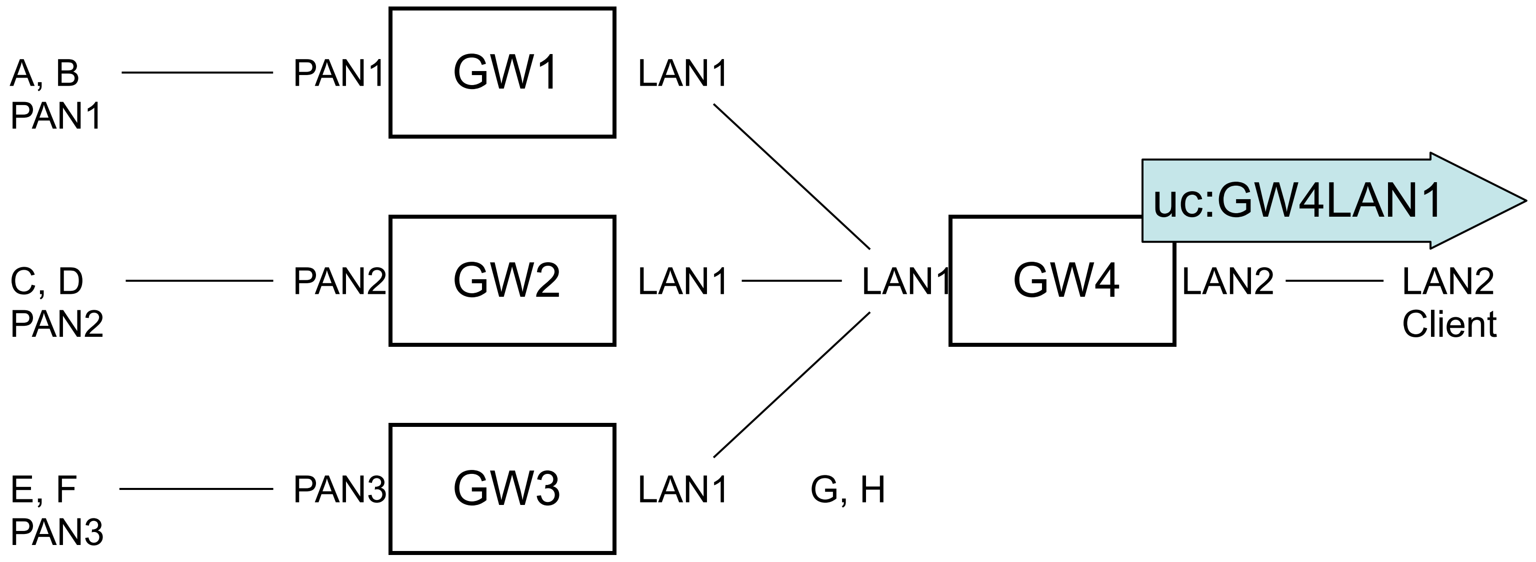
• (on-link)

Example: Gateway Discovery



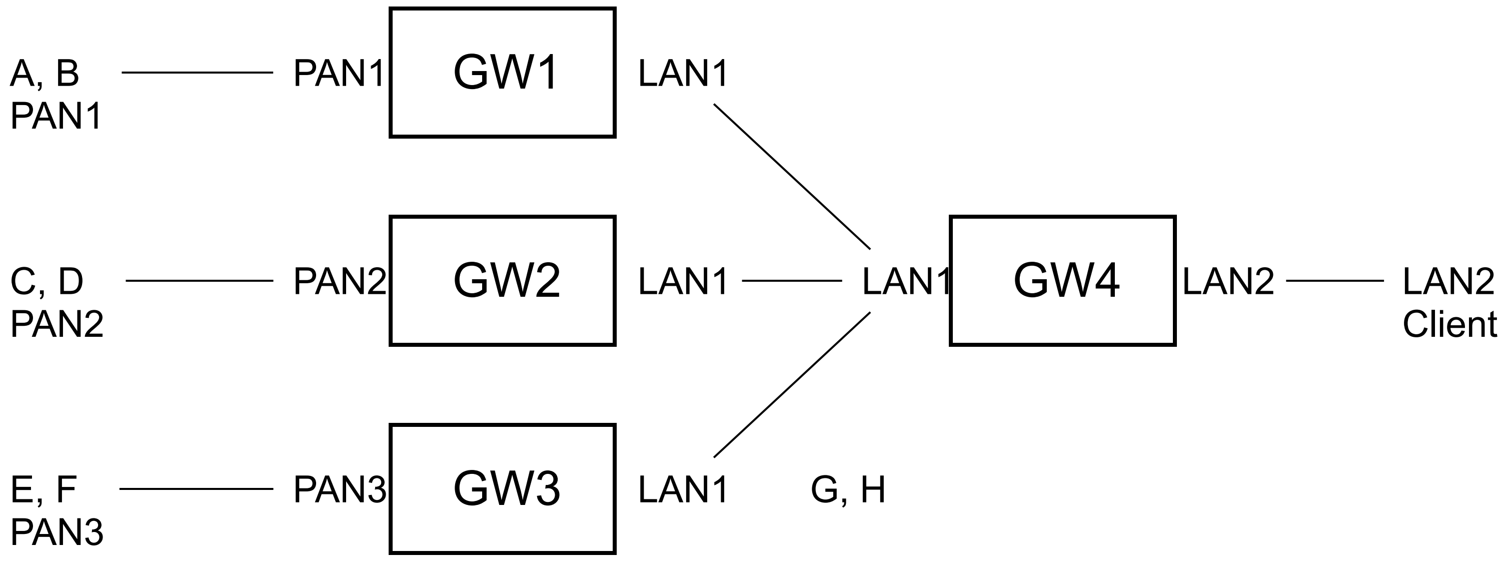
- (on-link)

Example: Gateway Discovery



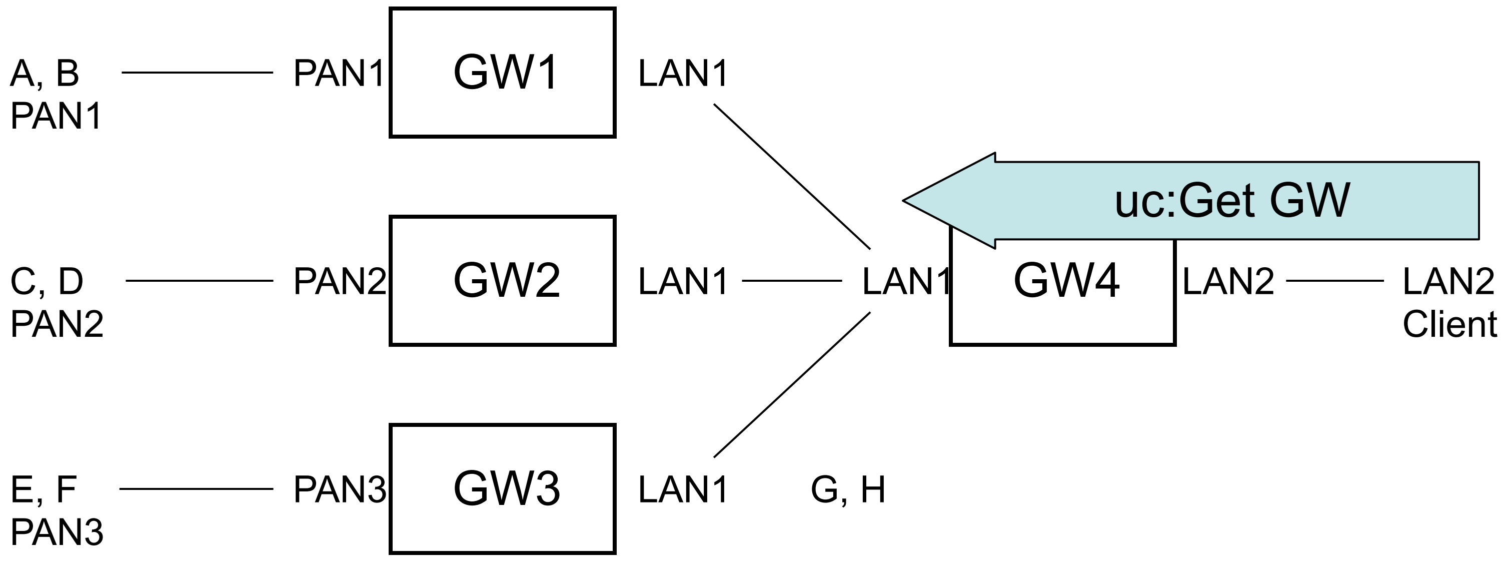
- (on-link)
- GW4LAN1

Example: Gateway Discovery



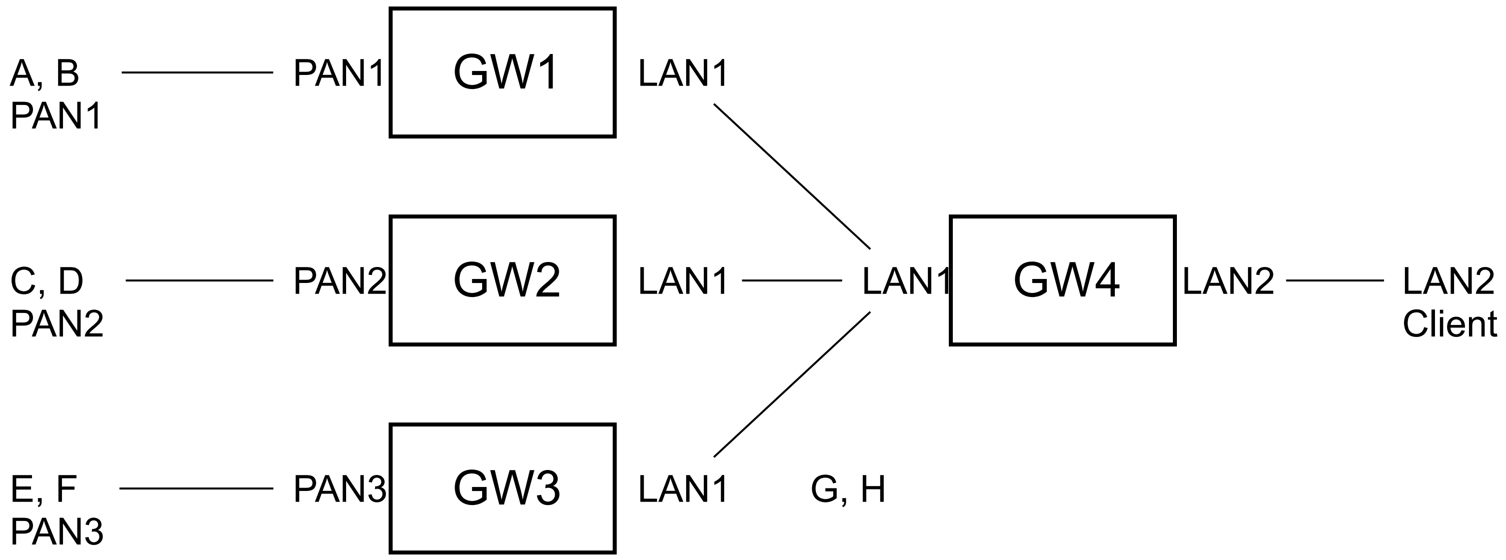
- (on-link)
- GW4LAN1

Example: Gateway Discovery



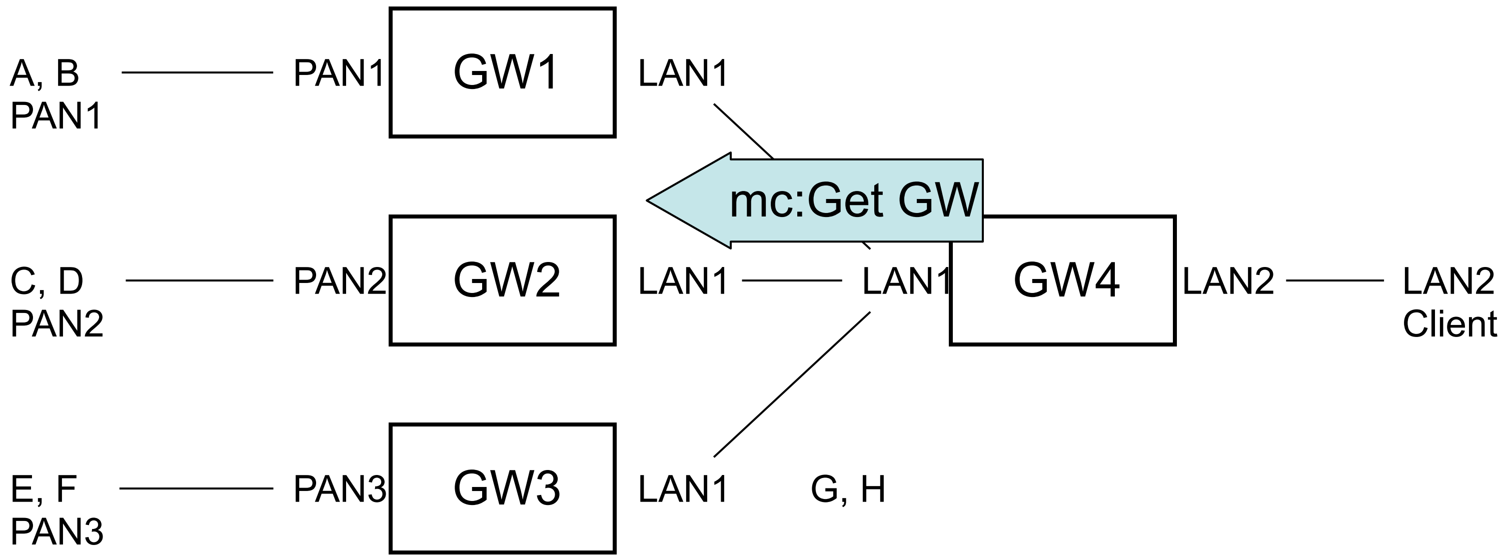
- (on-link)
- GW4LAN1

Example: Gateway Discovery



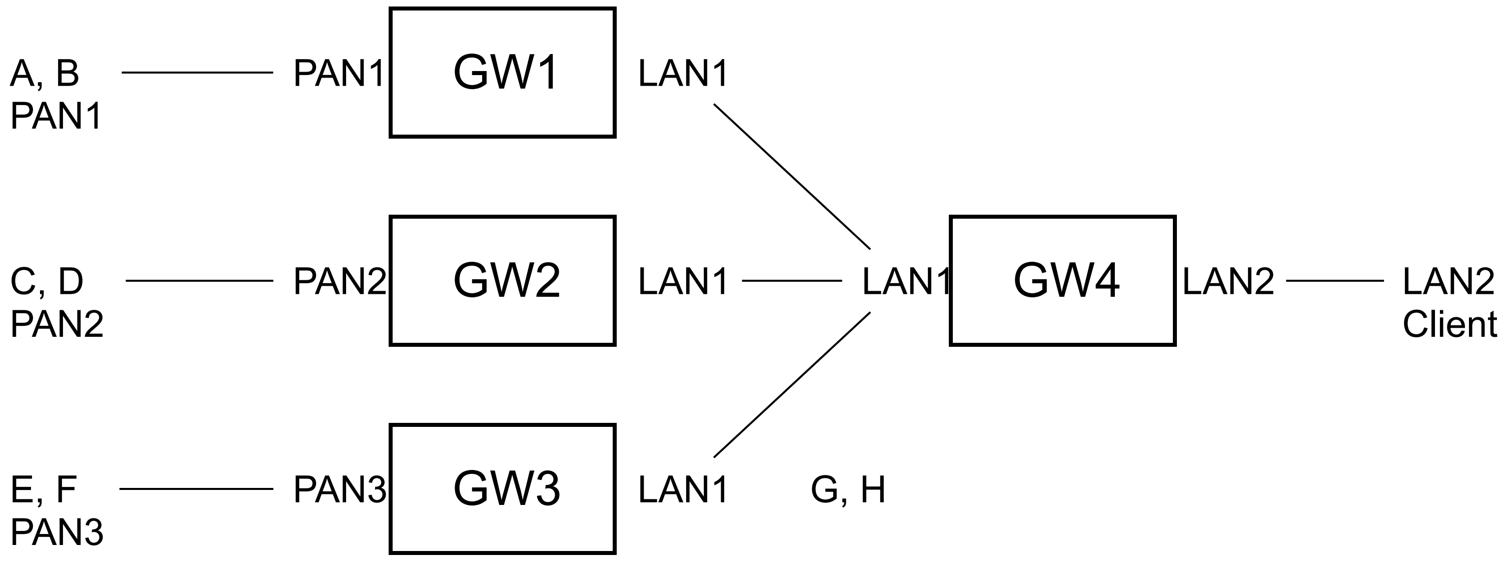
- (on-link)
- GW4LAN1

Example: Gateway Discovery



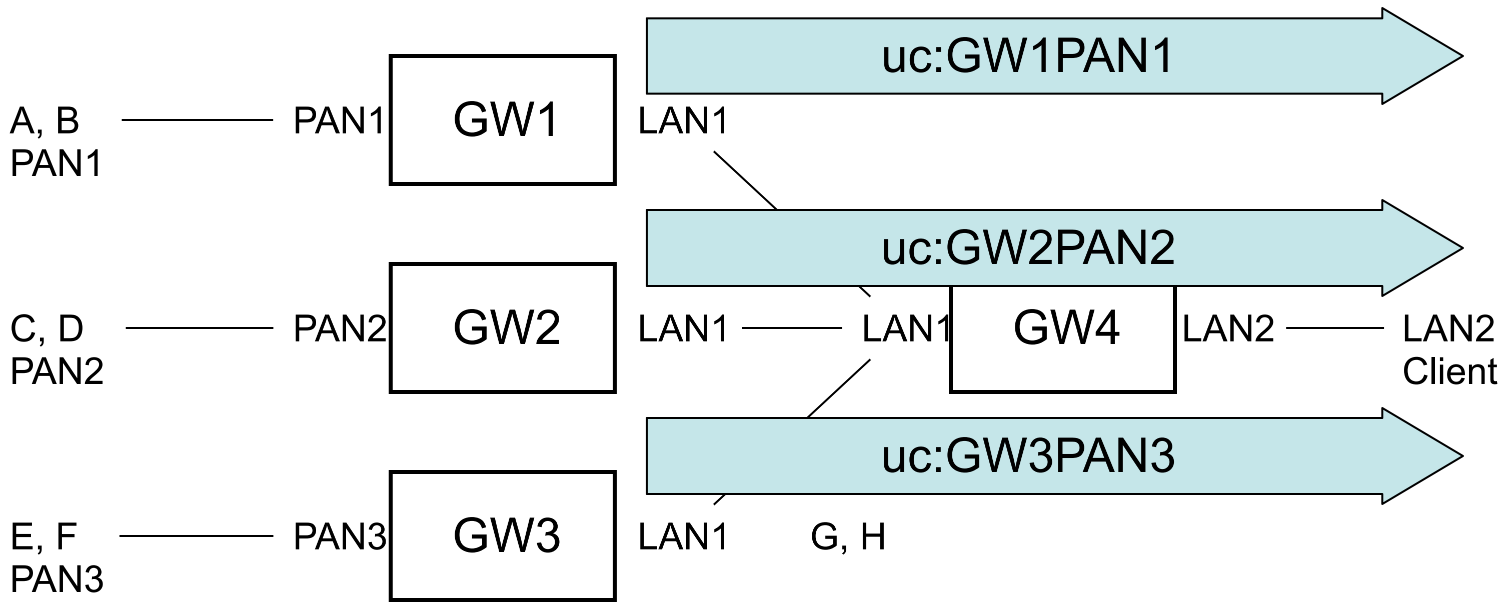
- (on-link)
- GW4LAN1

Example: Gateway Discovery



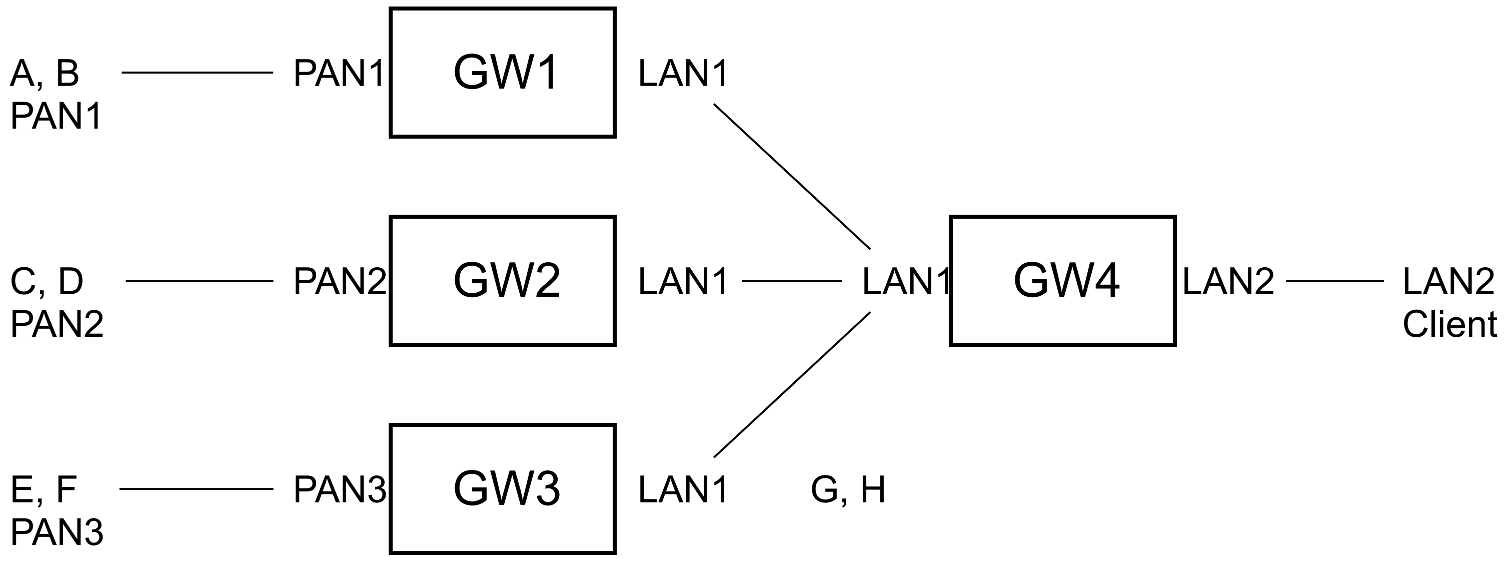
- (on-link)
- GW4LAN1

Example: Gateway Discovery



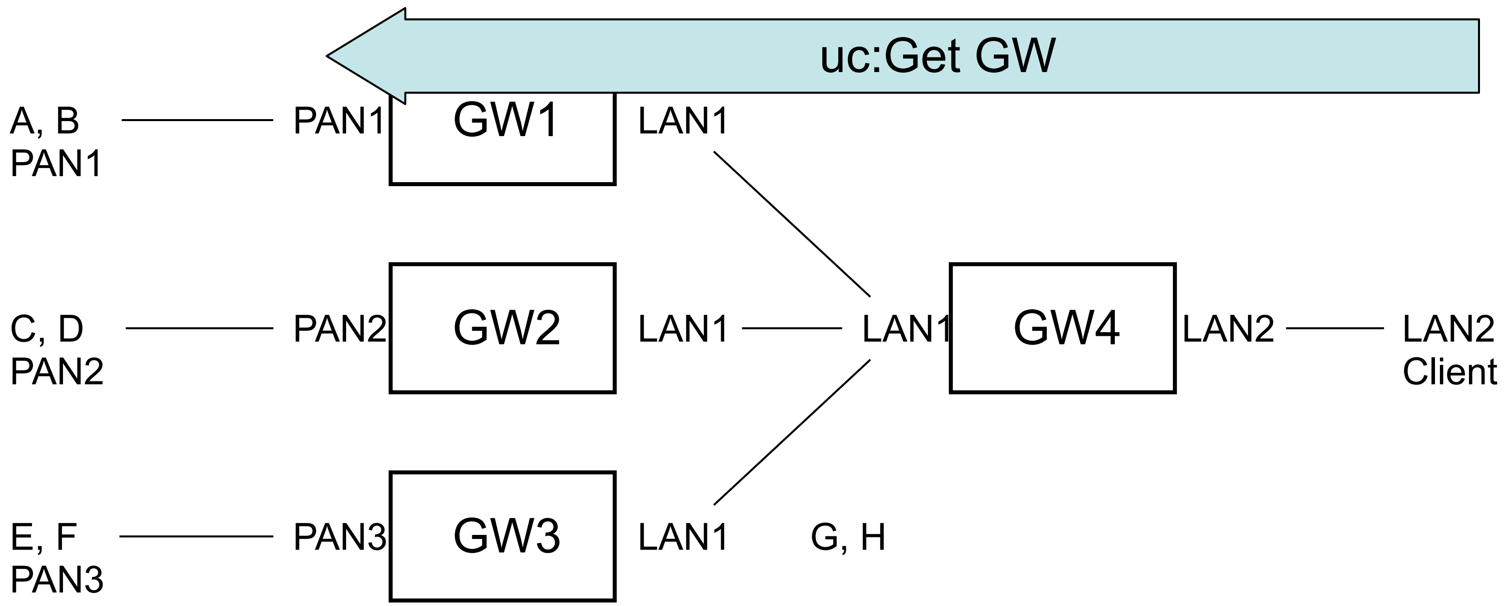
- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

Example: Gateway Discovery



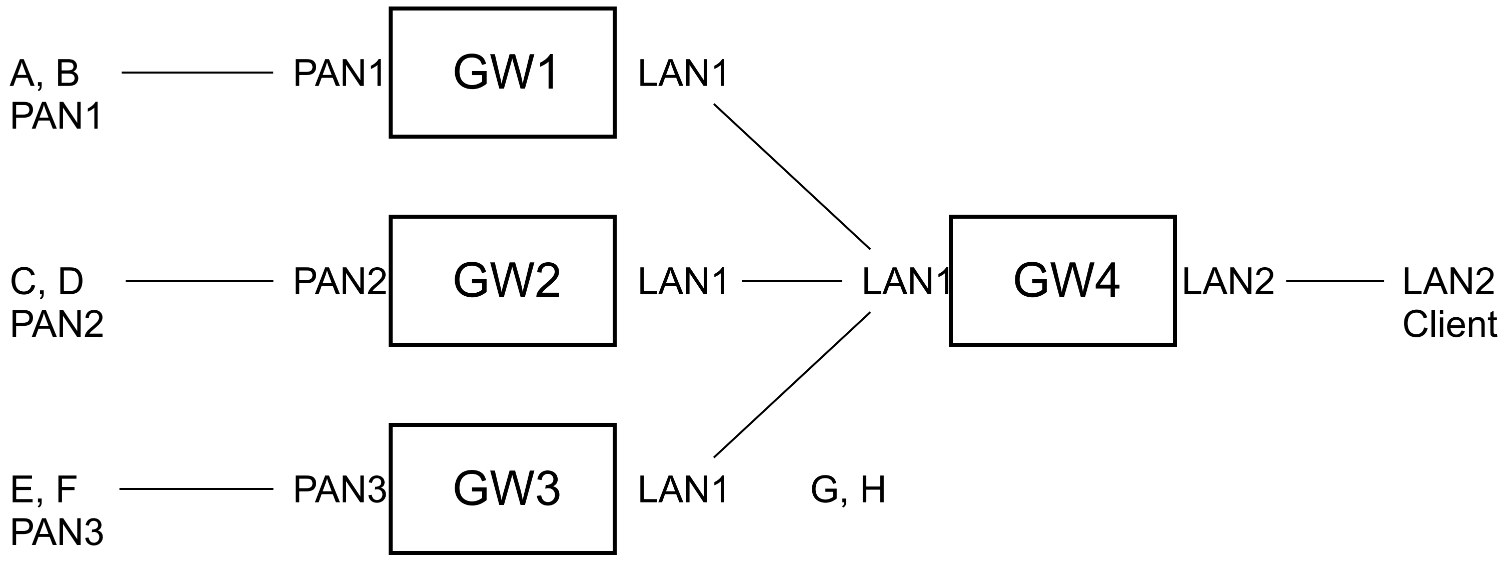
- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

Example: Gateway Discovery



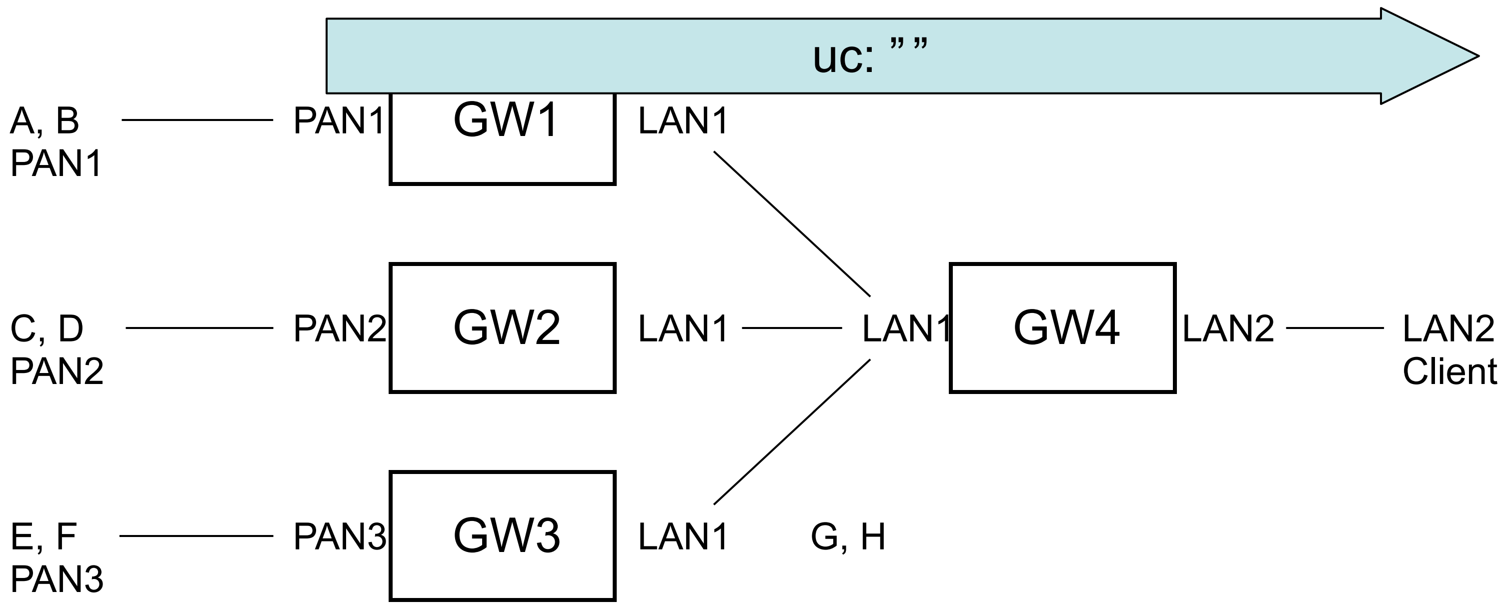
- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

Example: Gateway Discovery



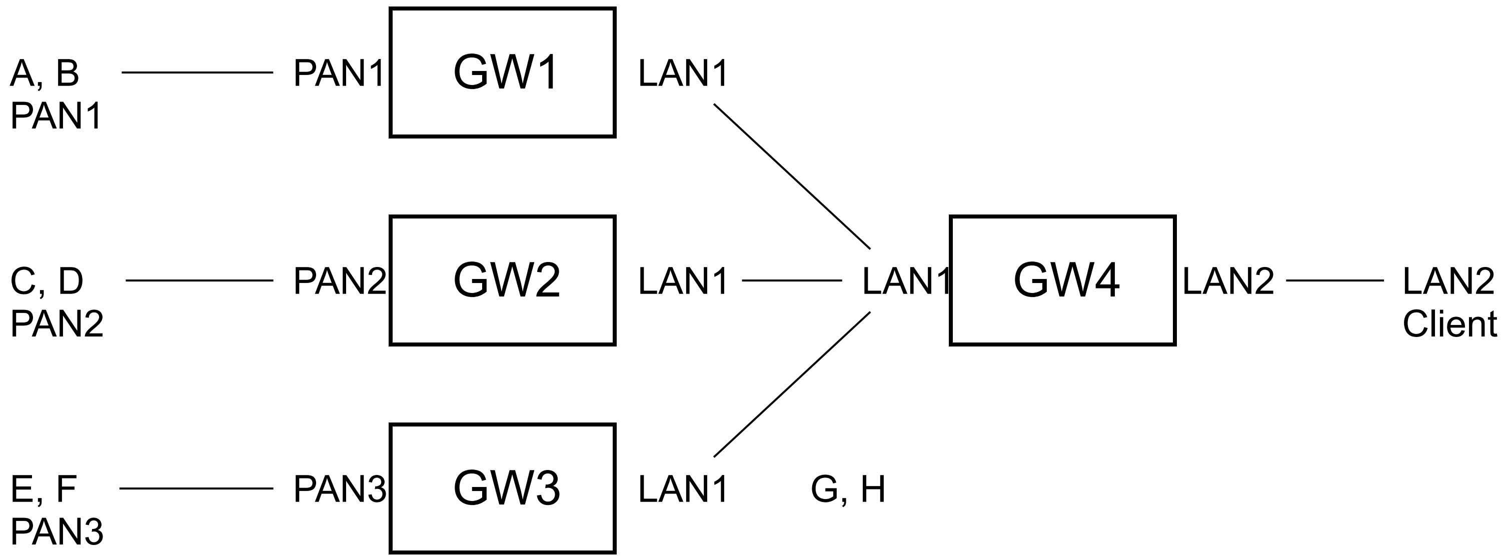
- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

Example: Gateway Discovery



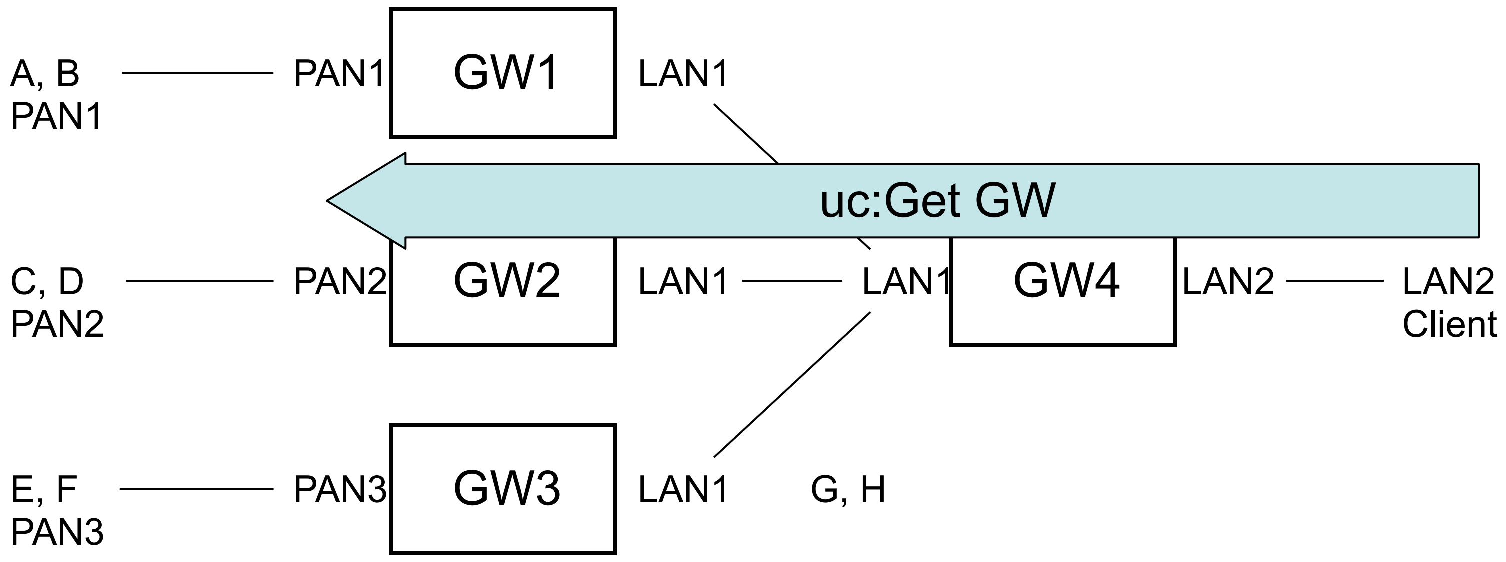
- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

Example: Gateway Discovery



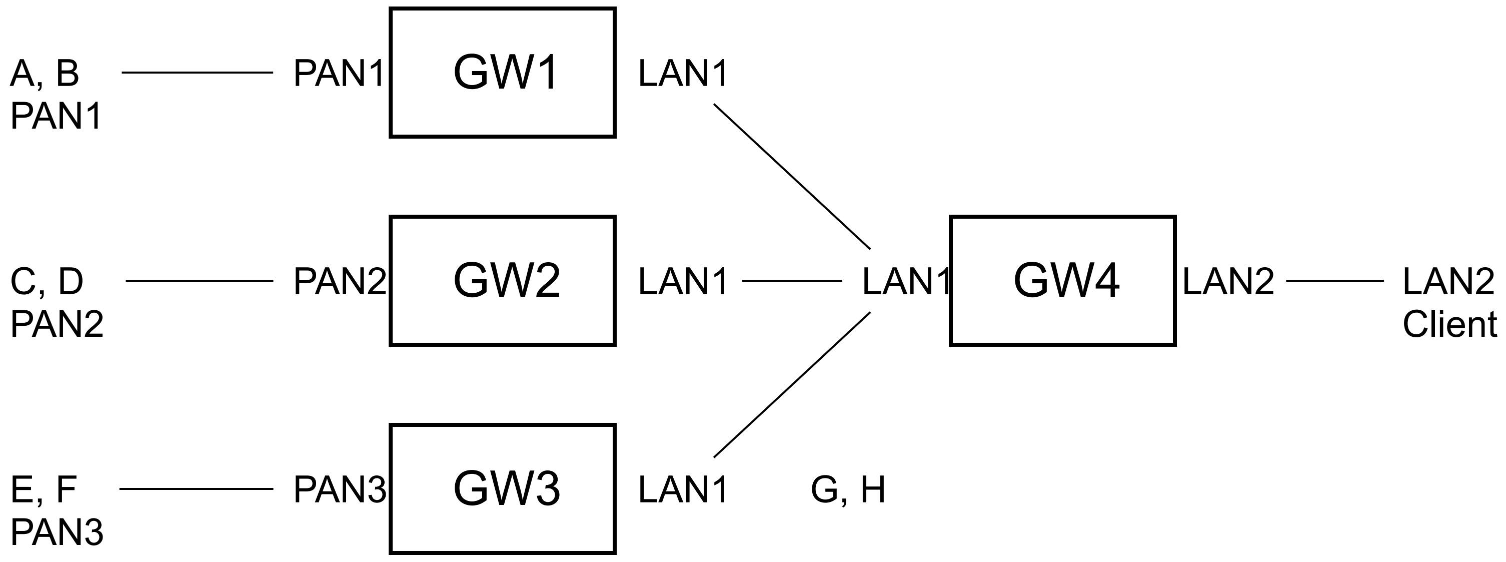
- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

Example: Gateway Discovery



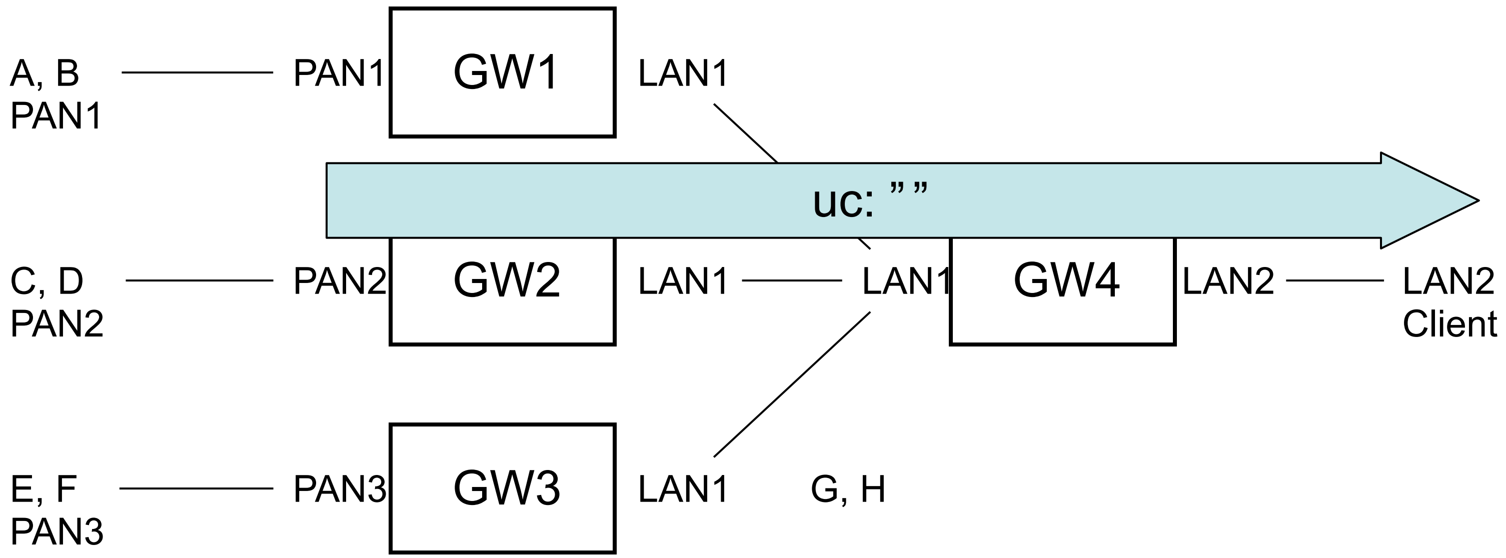
- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

Example: Gateway Discovery



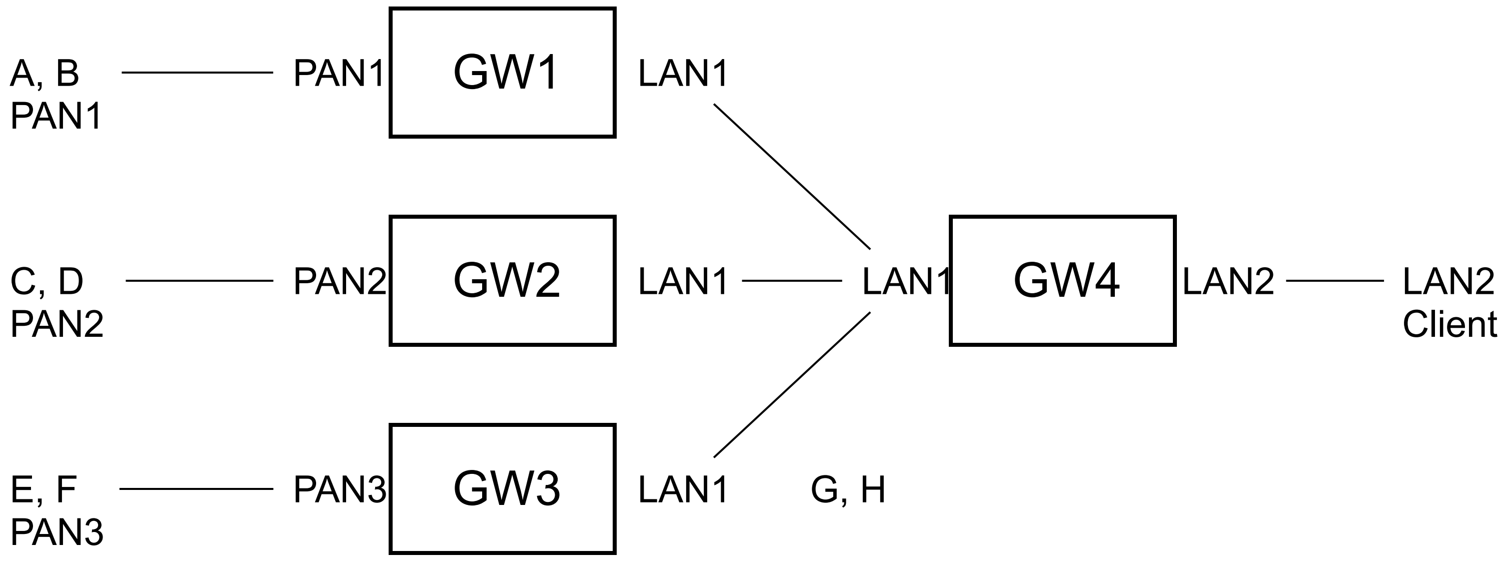
- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

Example: Gateway Discovery



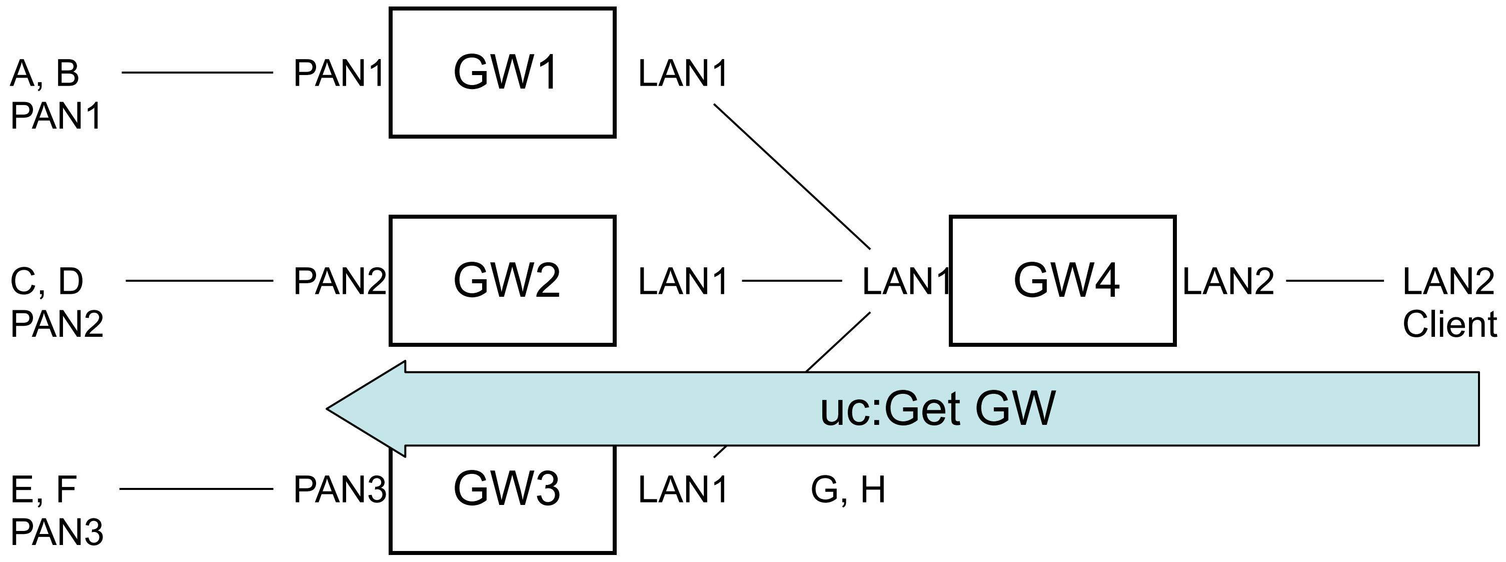
- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

Example: Gateway Discovery



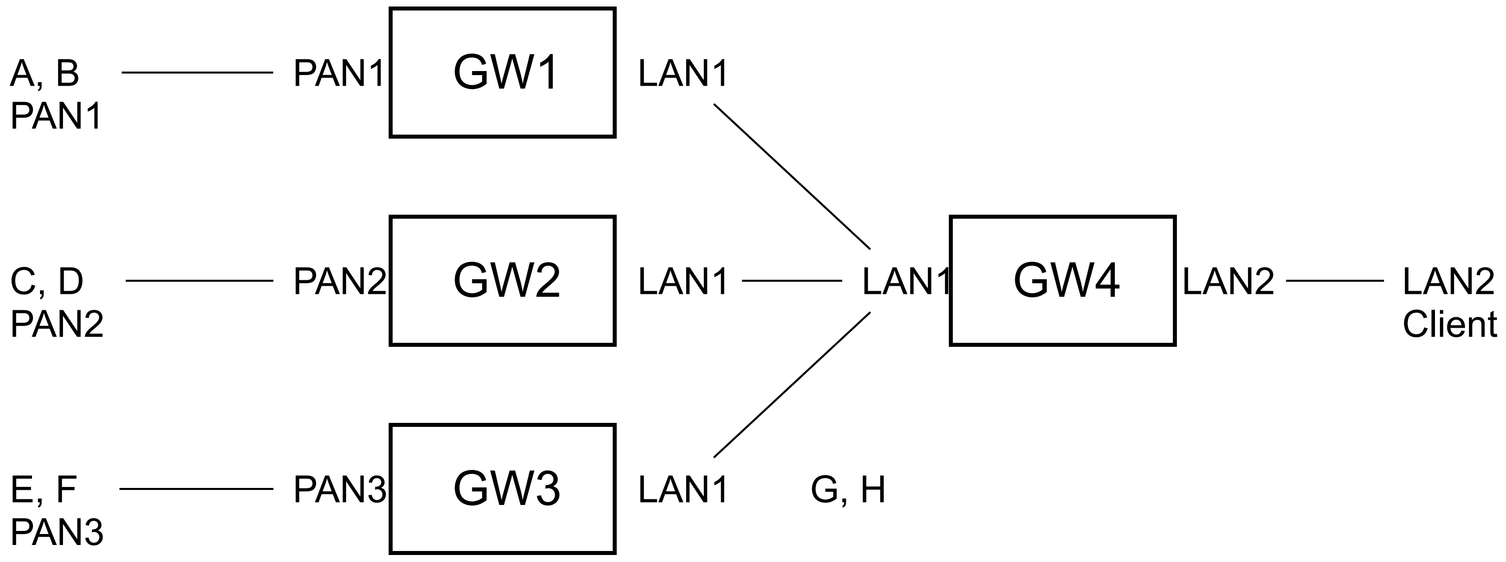
- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

Example: Gateway Discovery



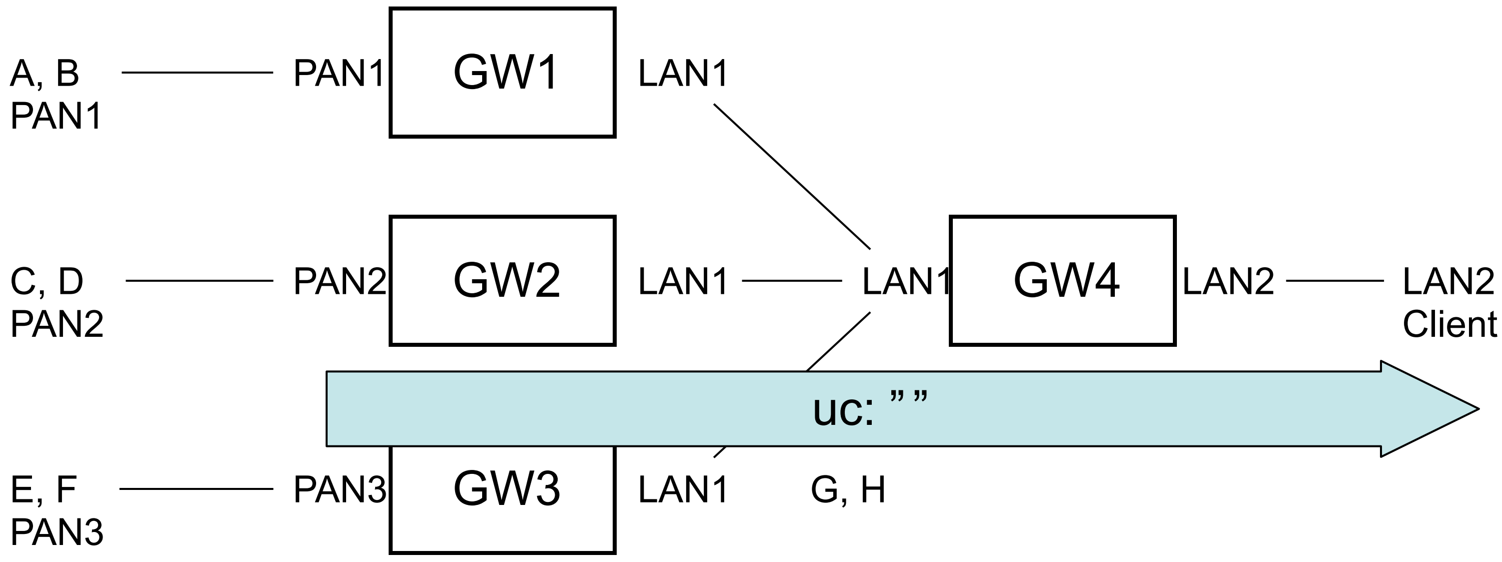
- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

Example: Gateway Discovery



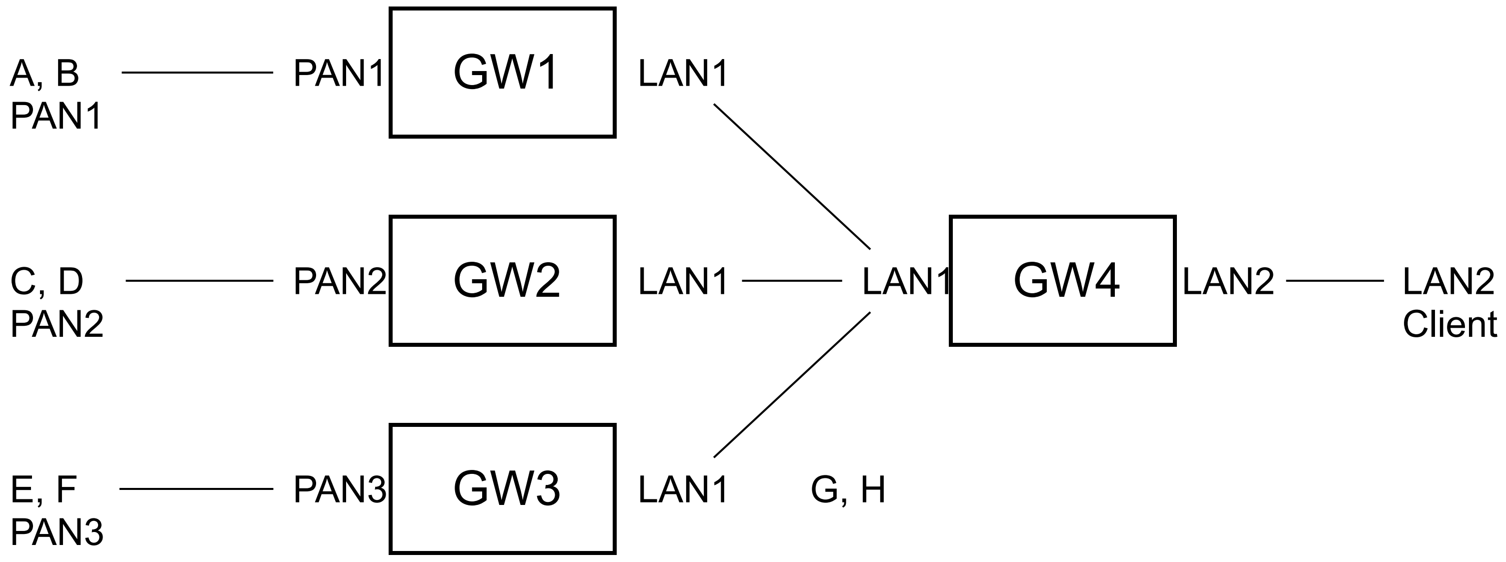
- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

Example: Gateway Discovery



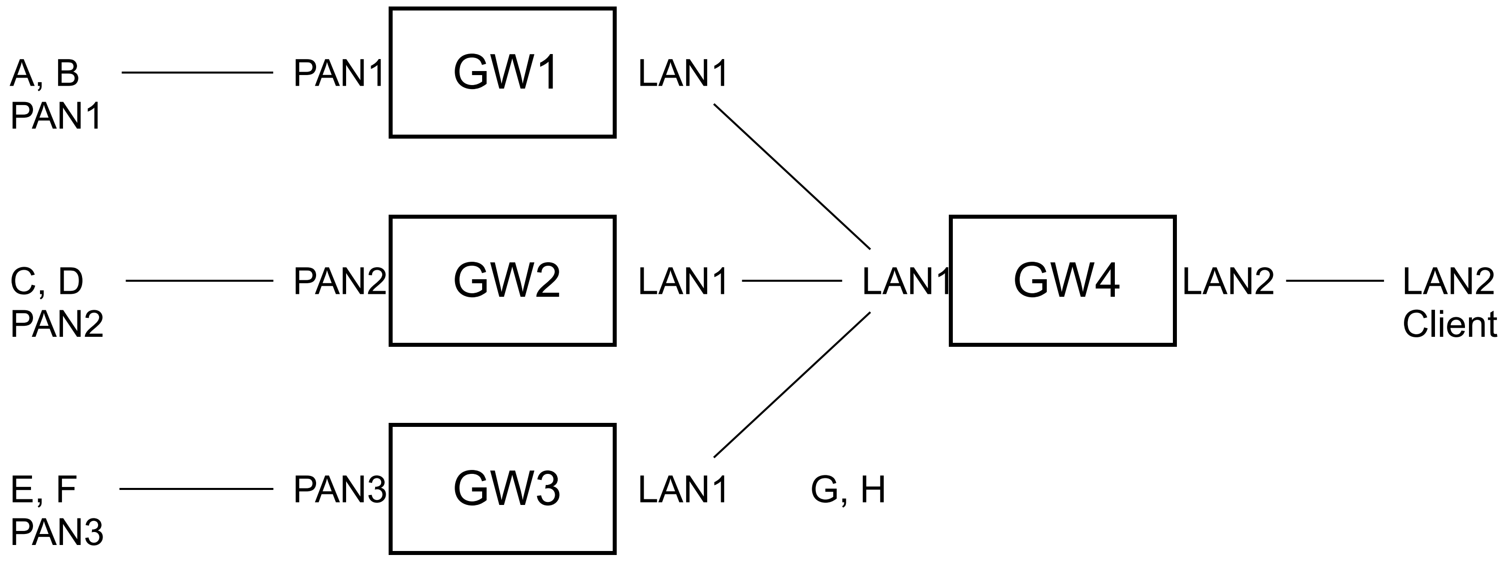
- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

Example: Gateway Discovery



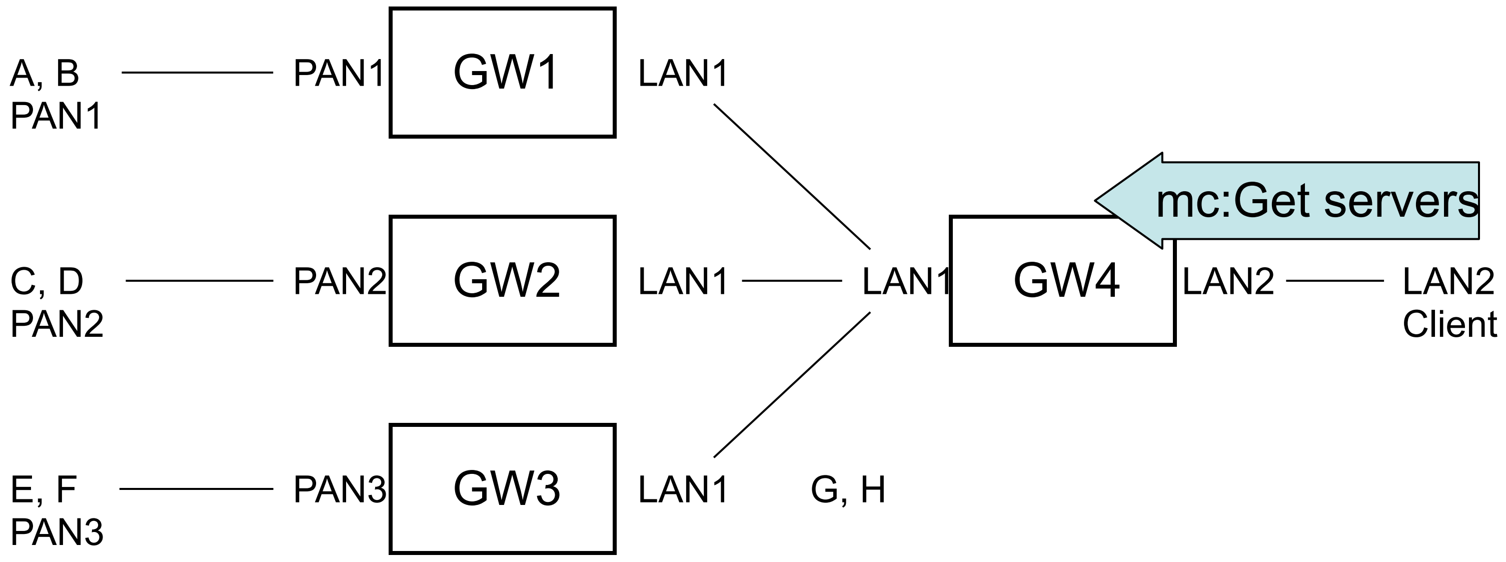
- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

Example: Server Discovery



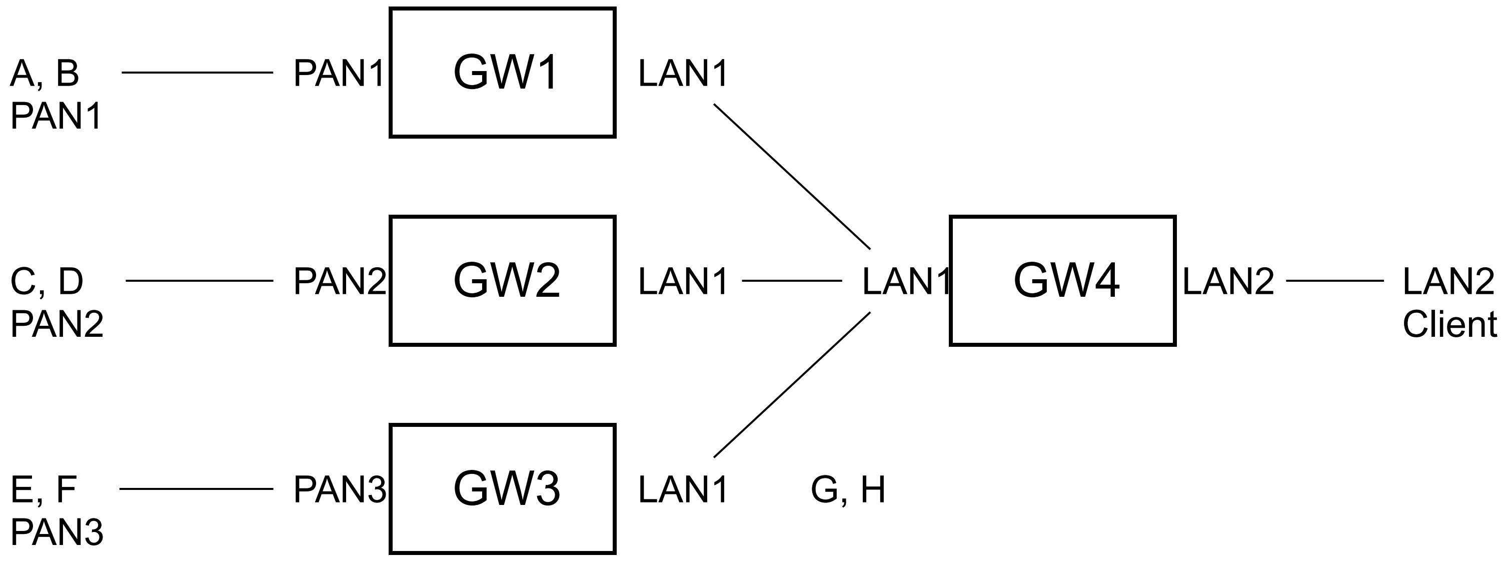
- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

Example: Server Discovery



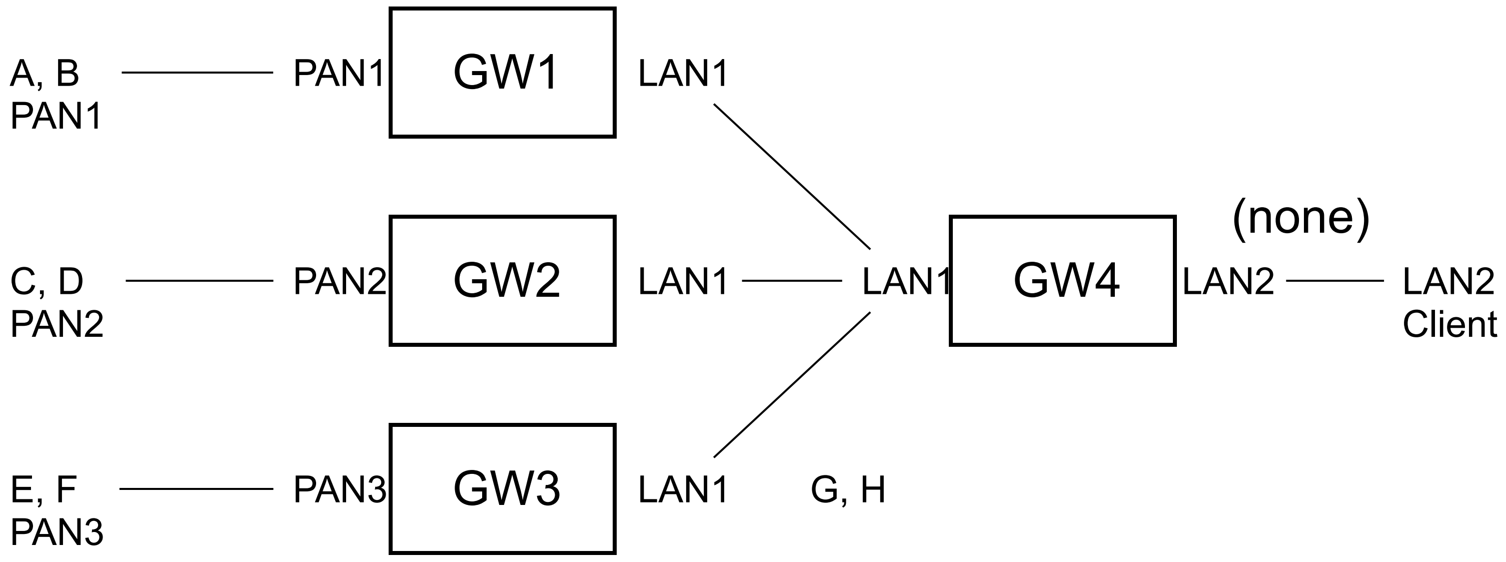
- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

Example: Server Discovery



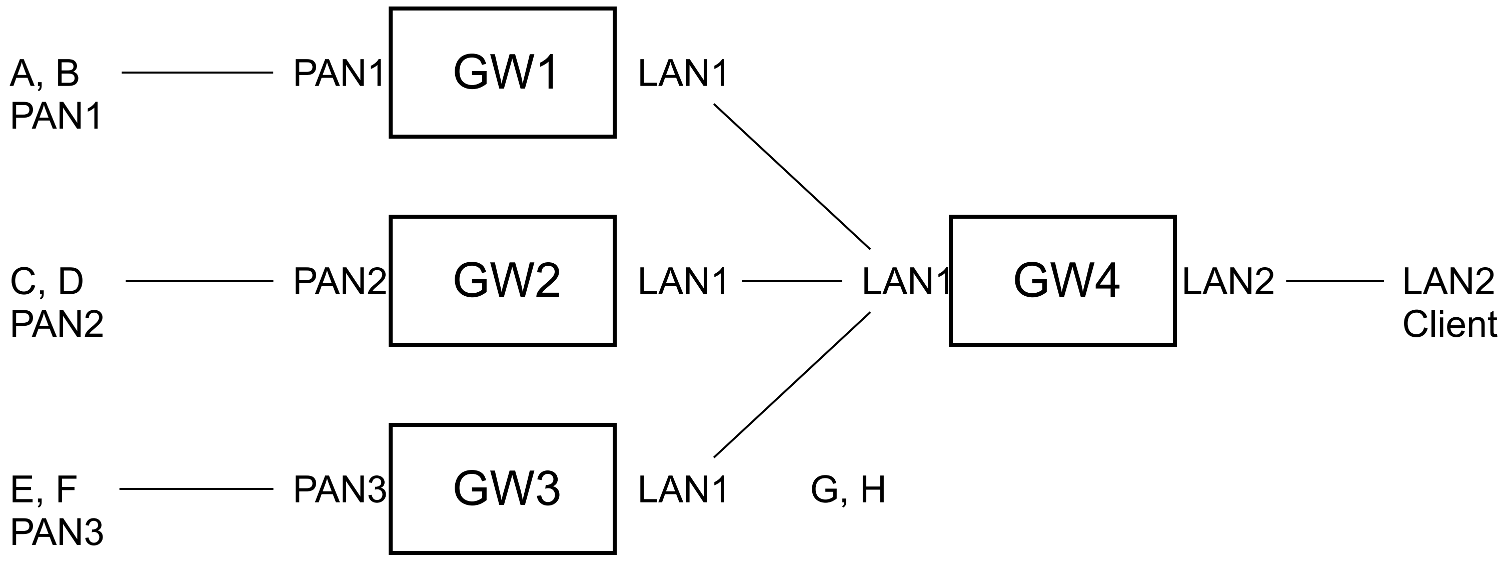
- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

Example: Server Discovery



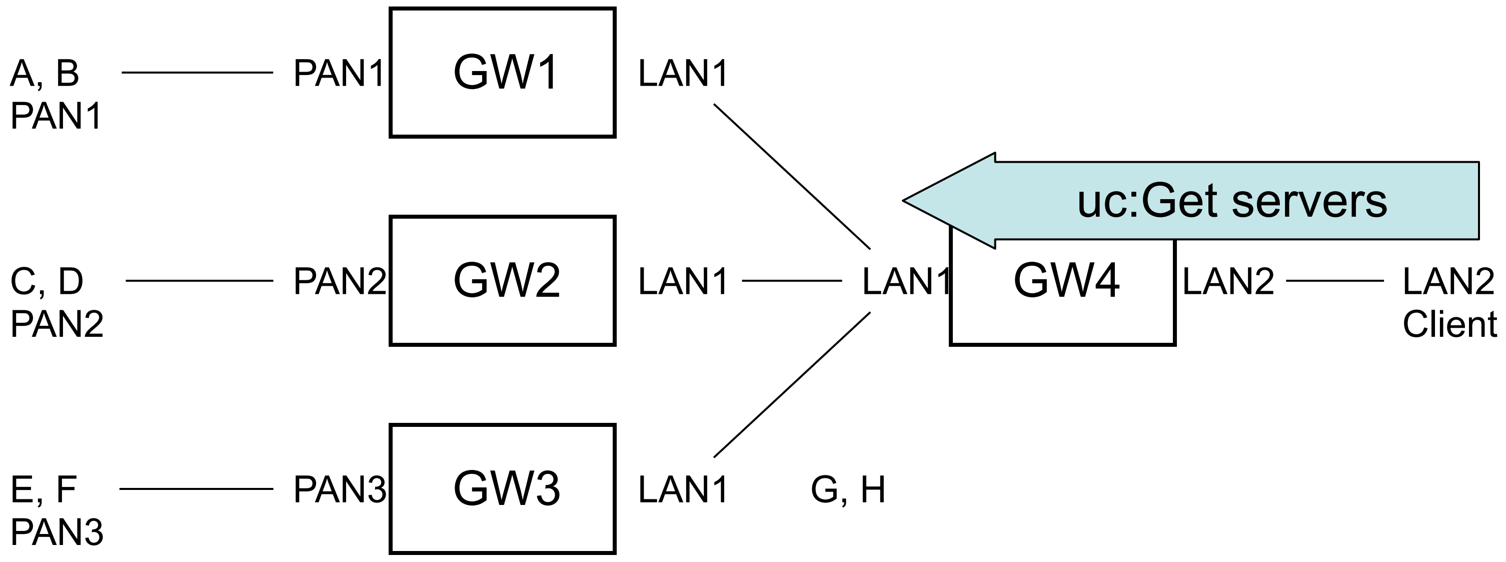
- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

Example: Server Discovery



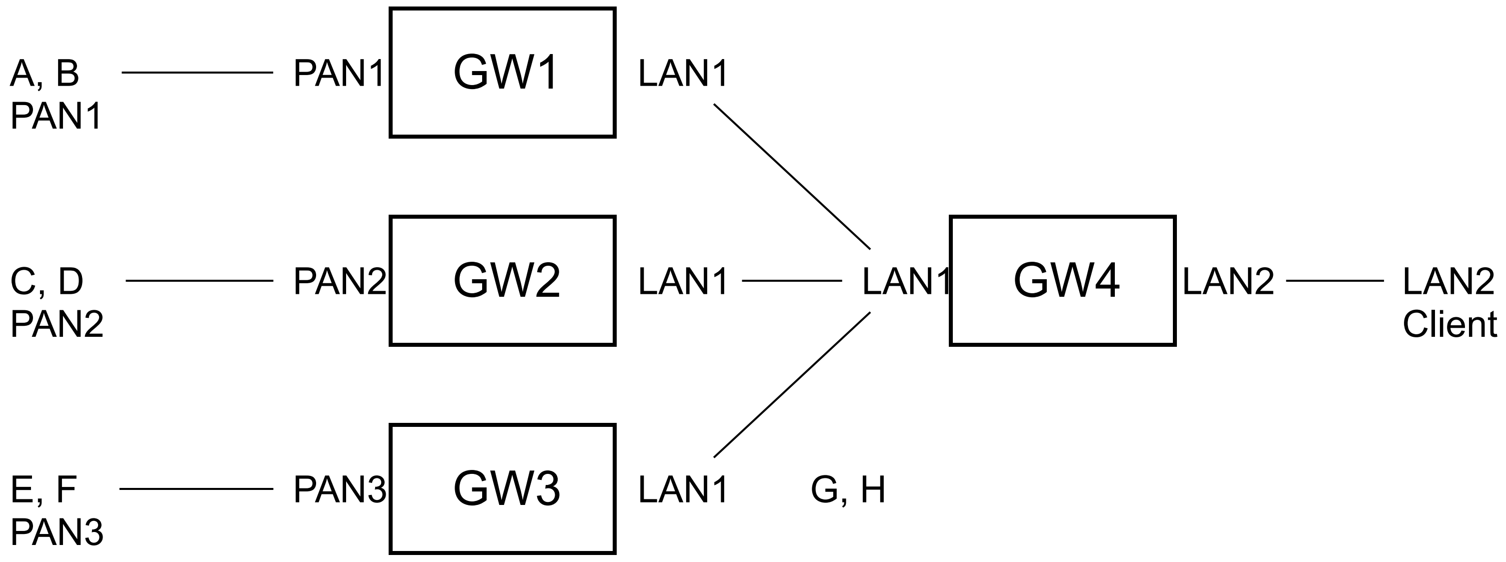
- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

Example: Server Discovery



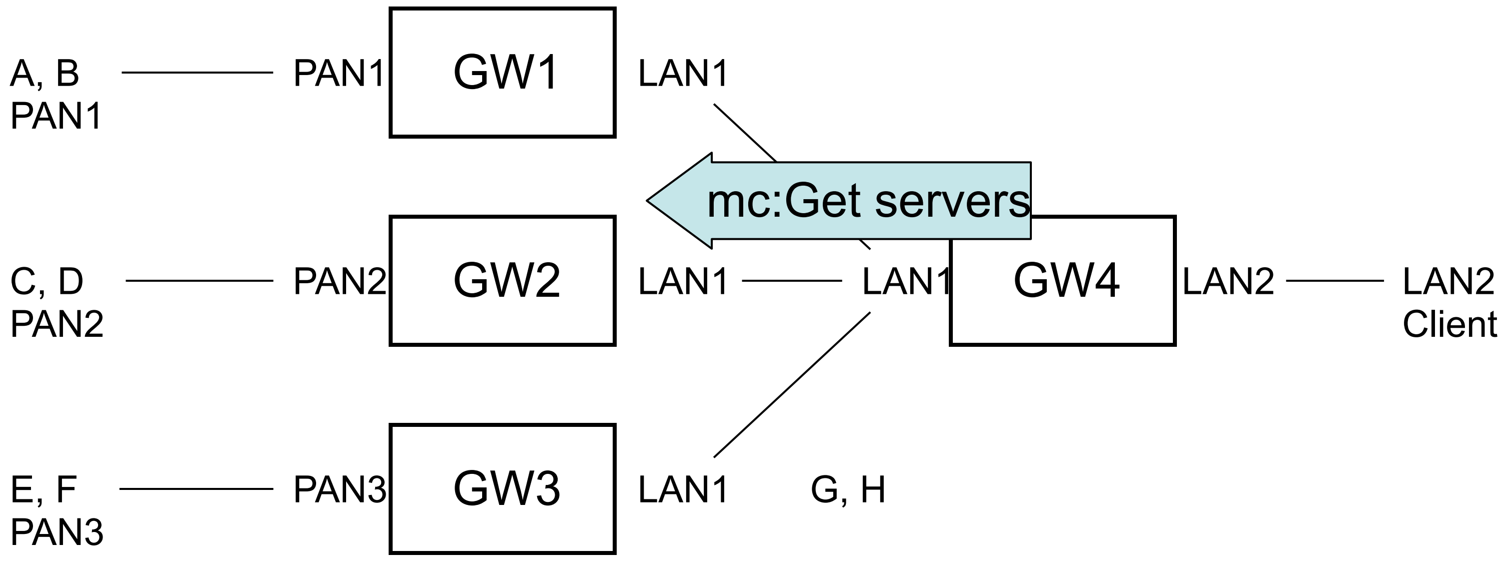
- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

Example: Server Discovery



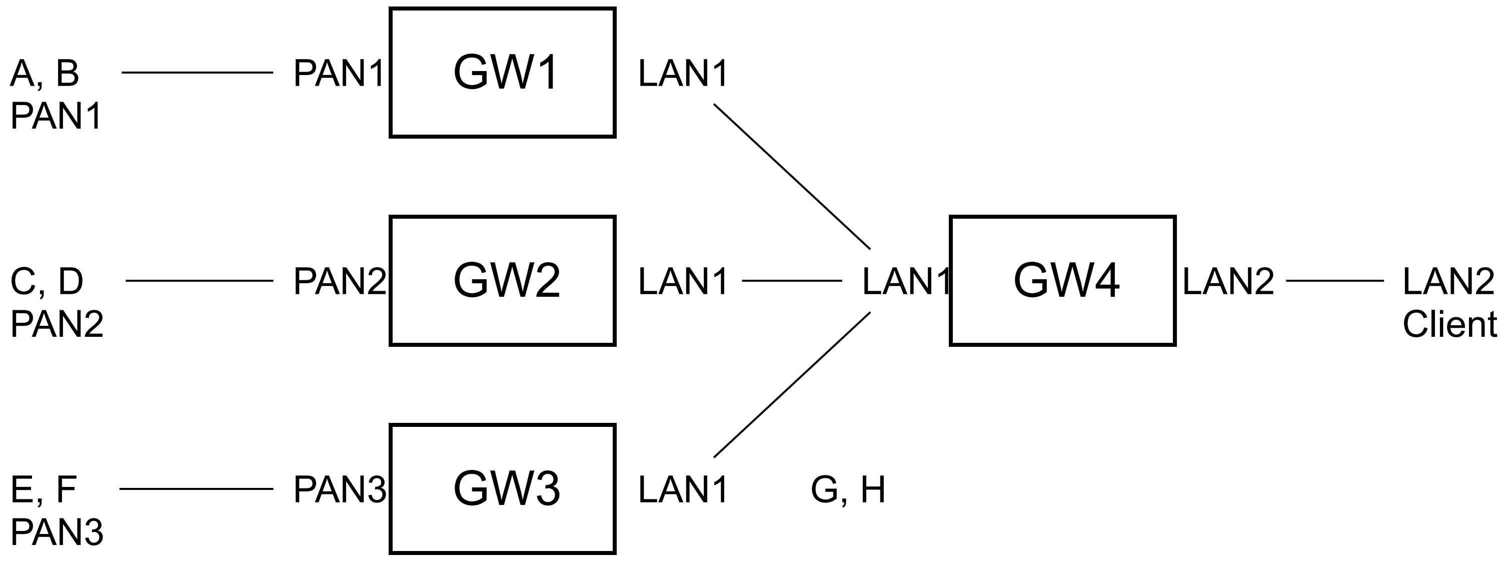
- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

Example: Server Discovery



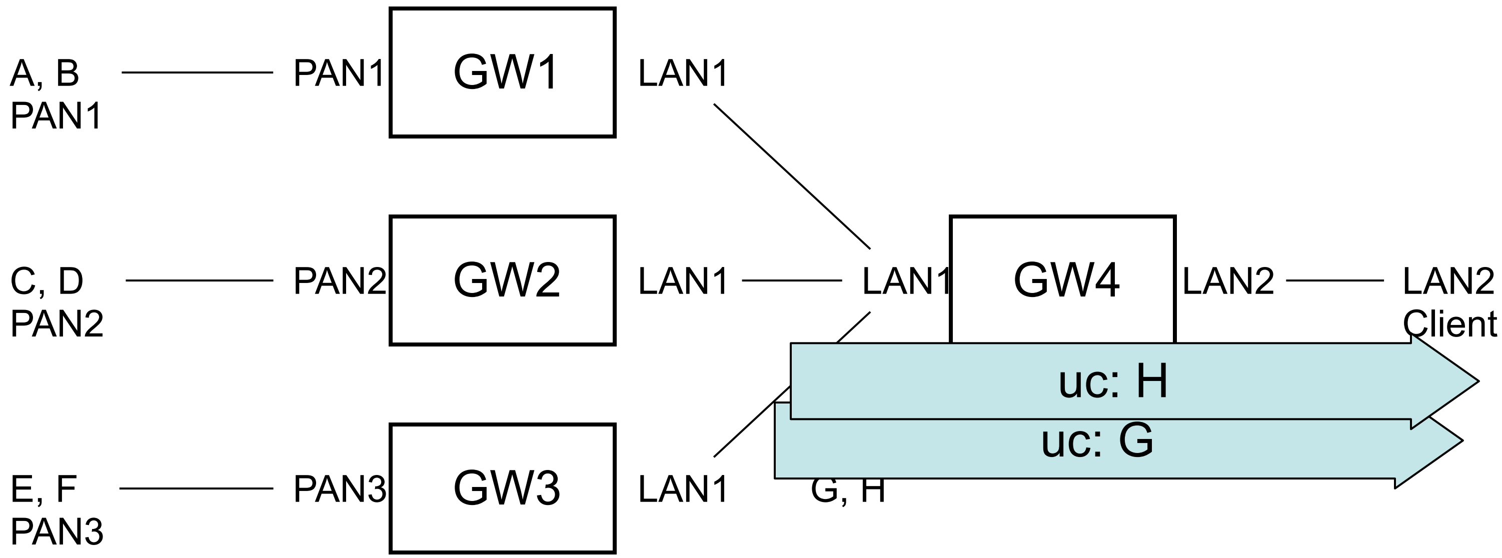
- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

Example: Server Discovery



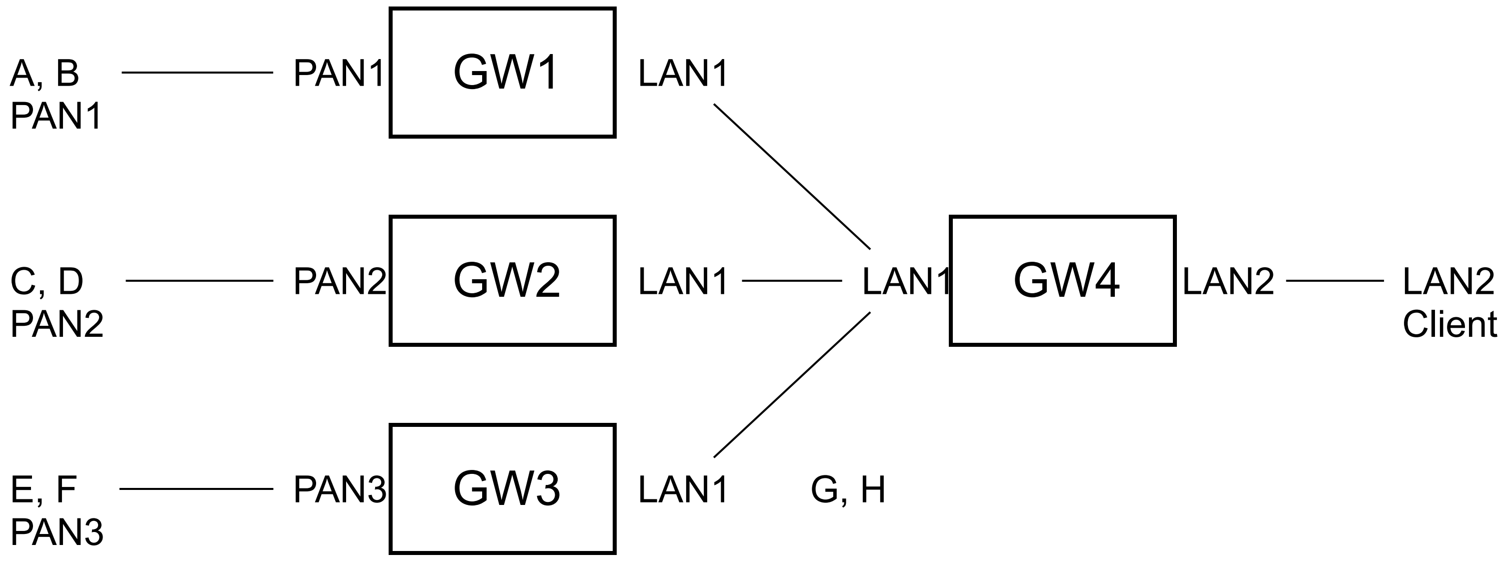
- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

Example: Server Discovery



- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

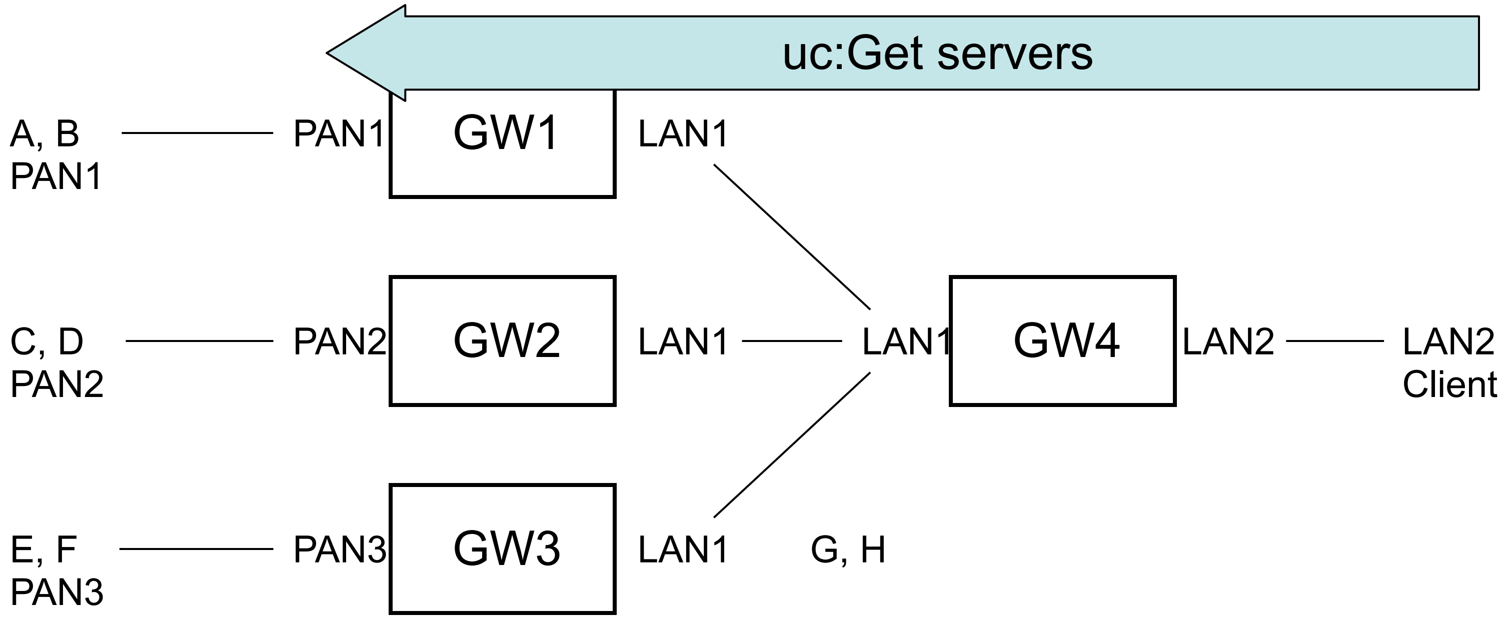
Example: Server Discovery



- G
- H

- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

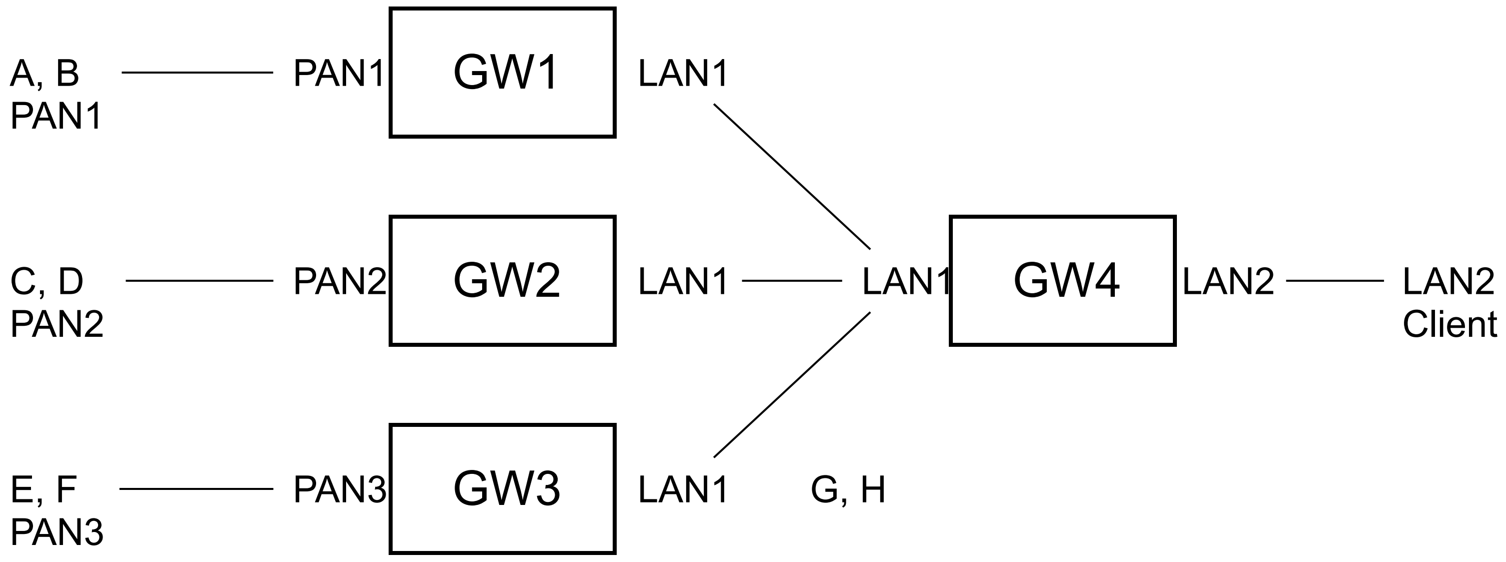
Example: Server Discovery



- G
- H

- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

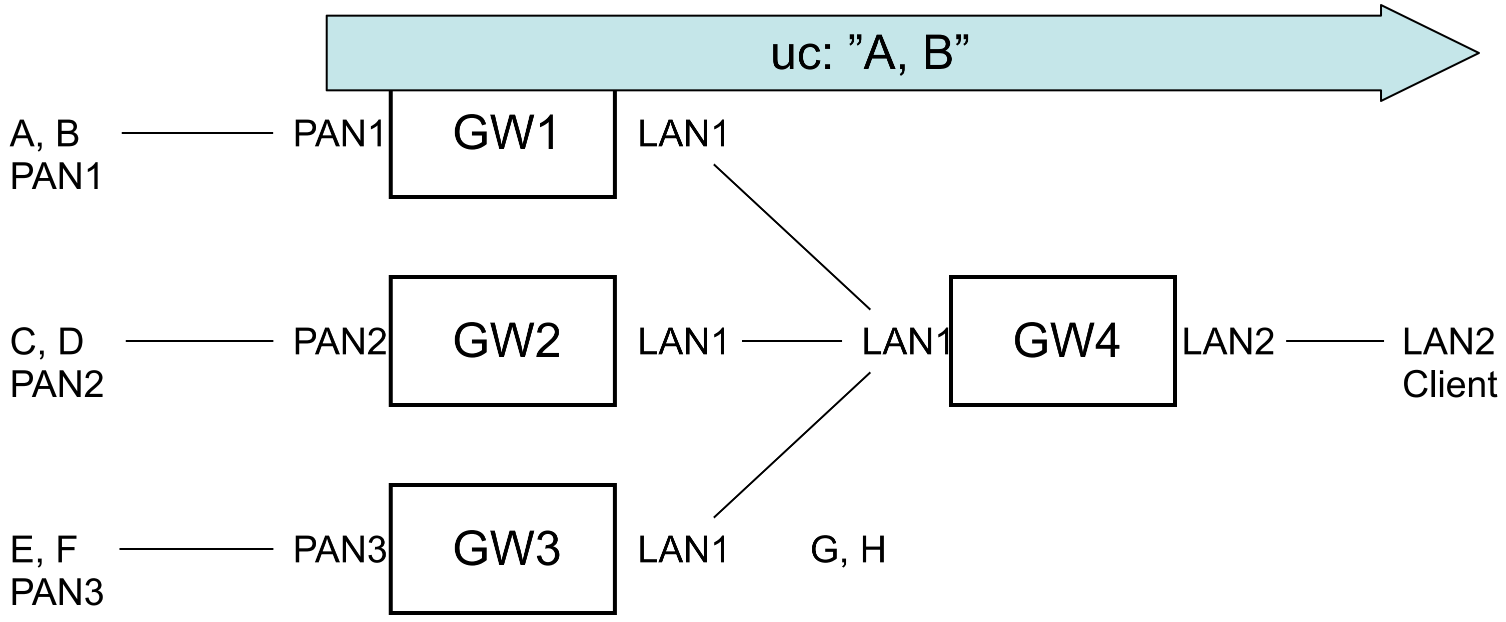
Example: Server Discovery



- G
- H

- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

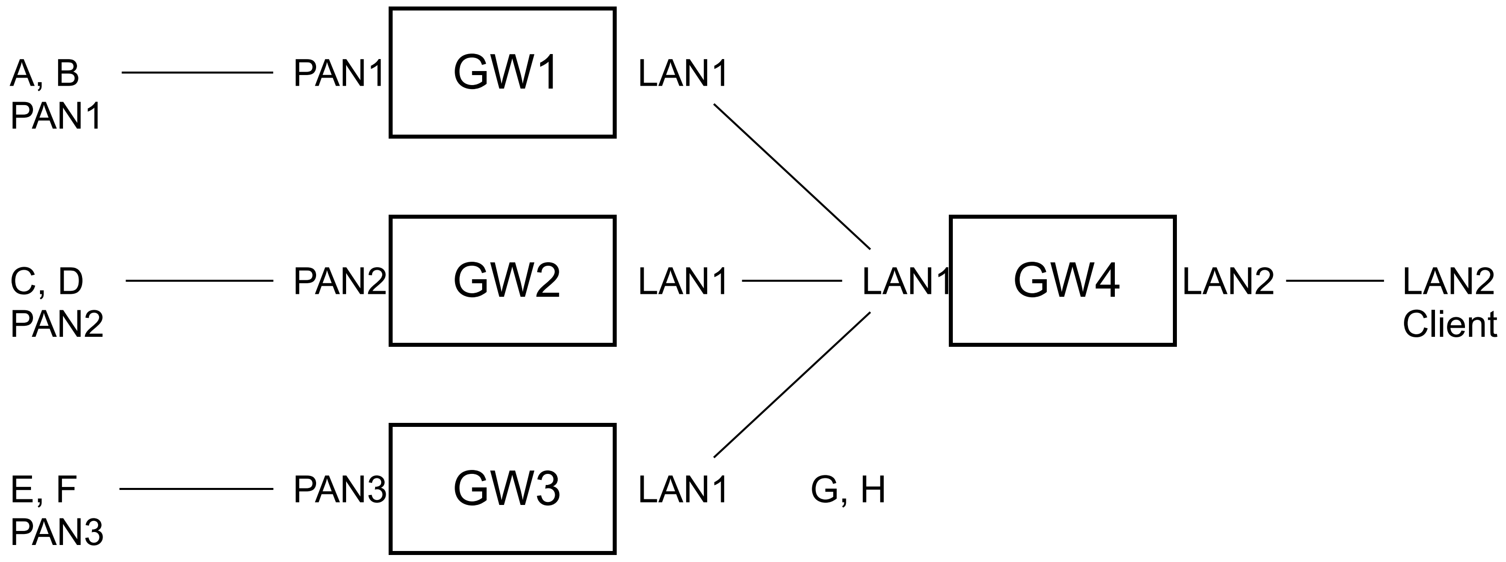
Example: Server Discovery



- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

- G
- H
- A
- B

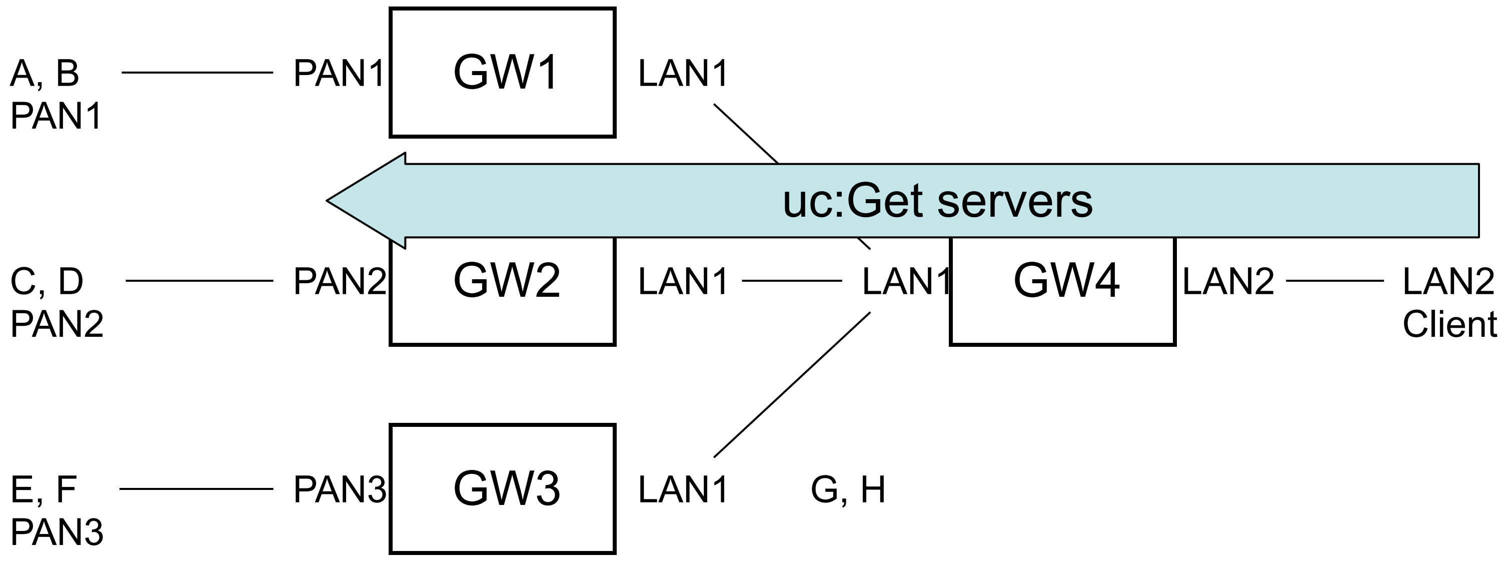
Example: Server Discovery



- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

- G
- H
- A
- B

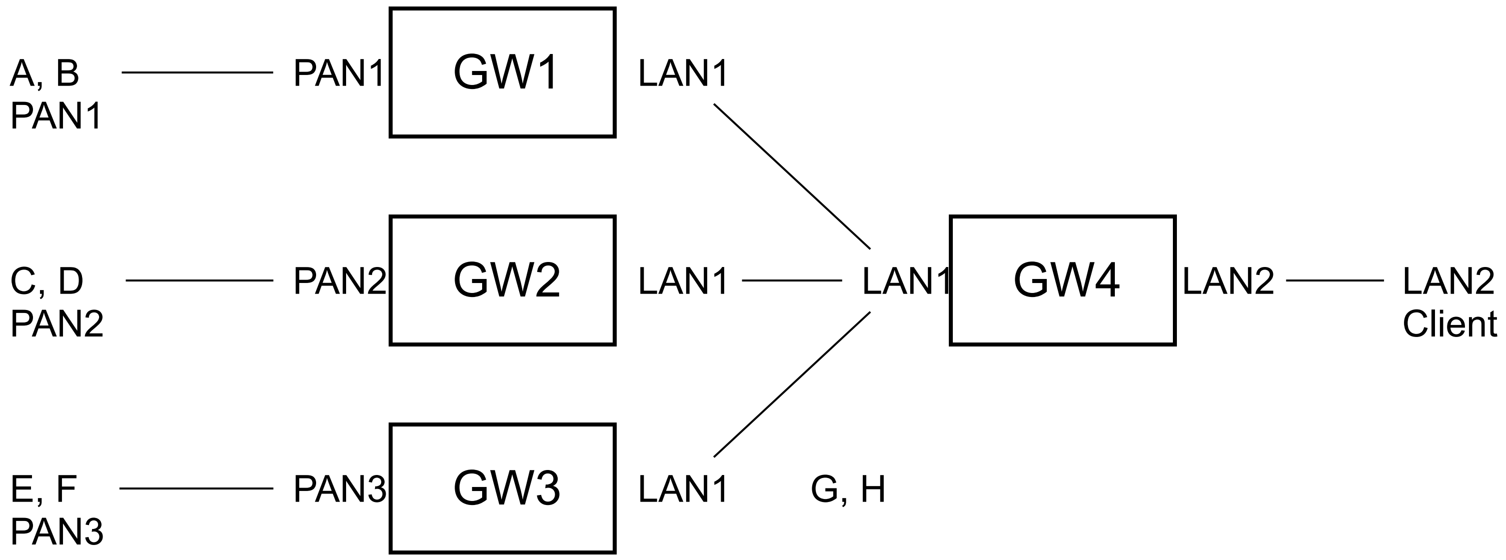
Example: Server Discovery



- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

- G
- H
- A
- B

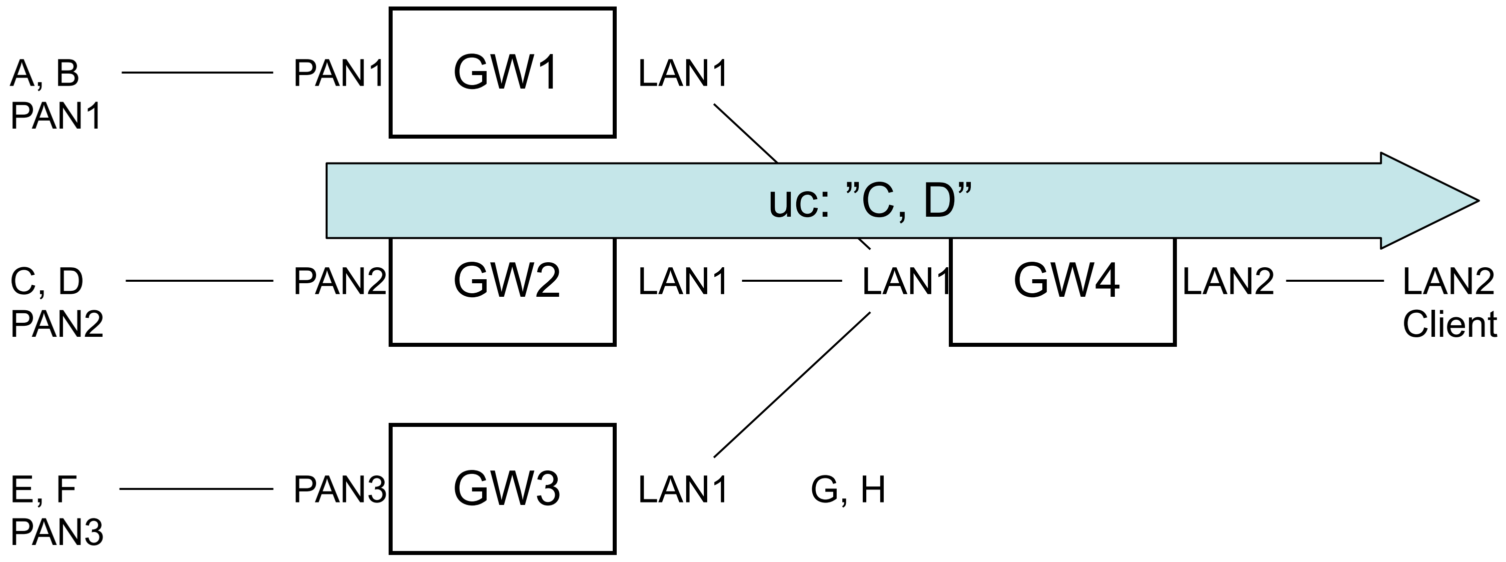
Example: Server Discovery



- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

- G
- H
- A
- B

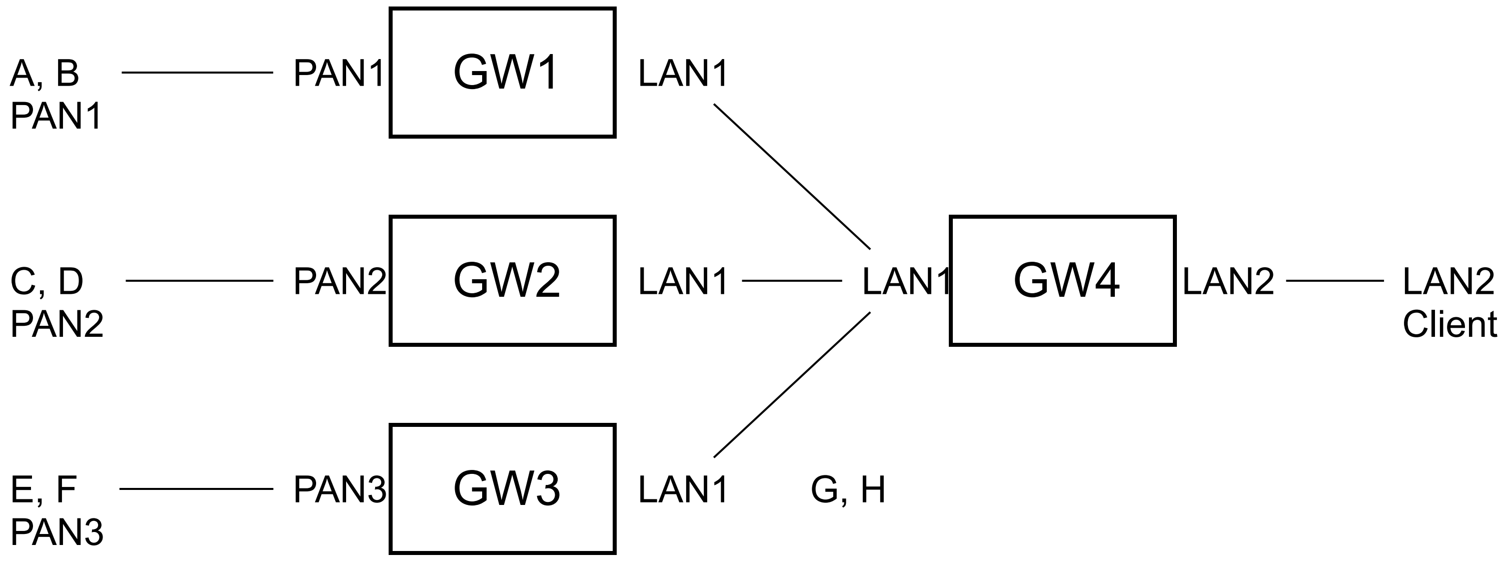
Example: Server Discovery



- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

- G
- H
- A
- B
- C
- D

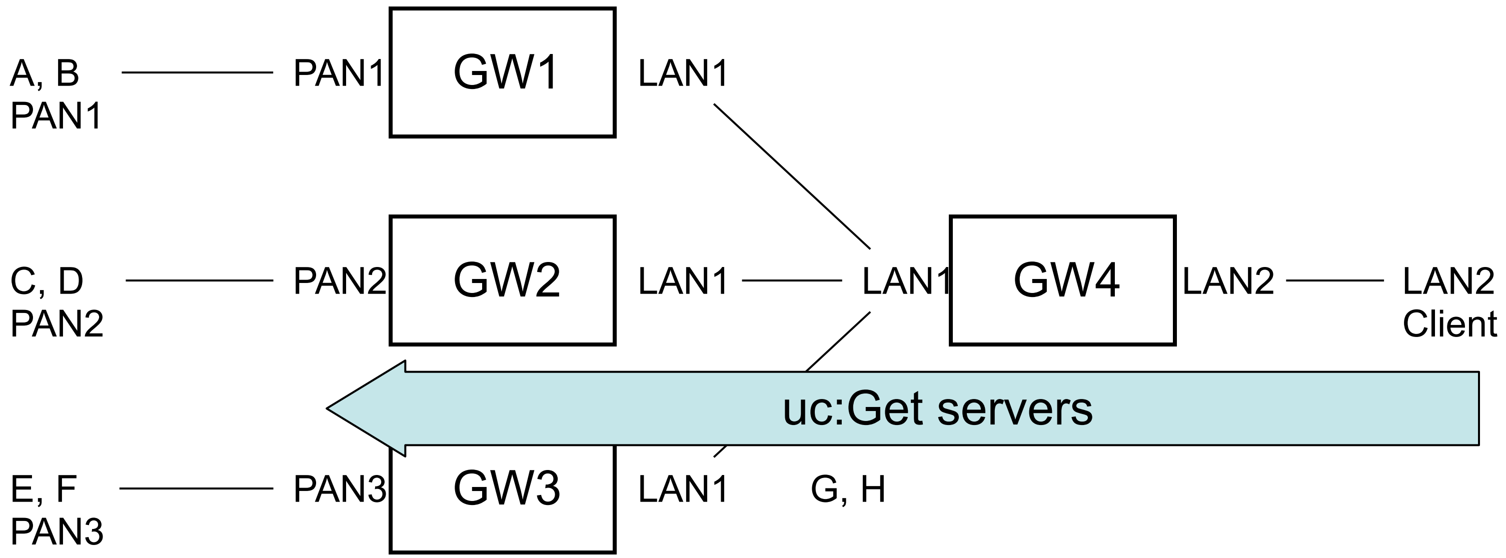
Example: Server Discovery



- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

- G
- H
- A
- B
- C
- D

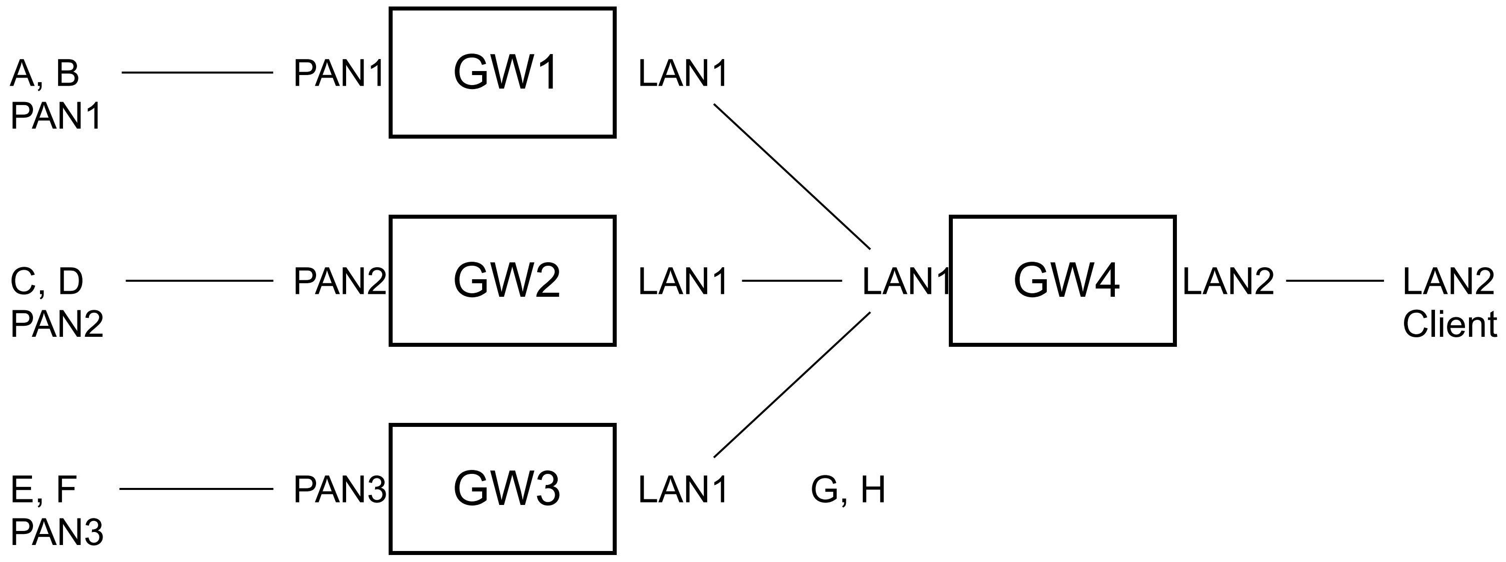
Example: Server Discovery



- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

- G
- H
- A
- B
- C
- D

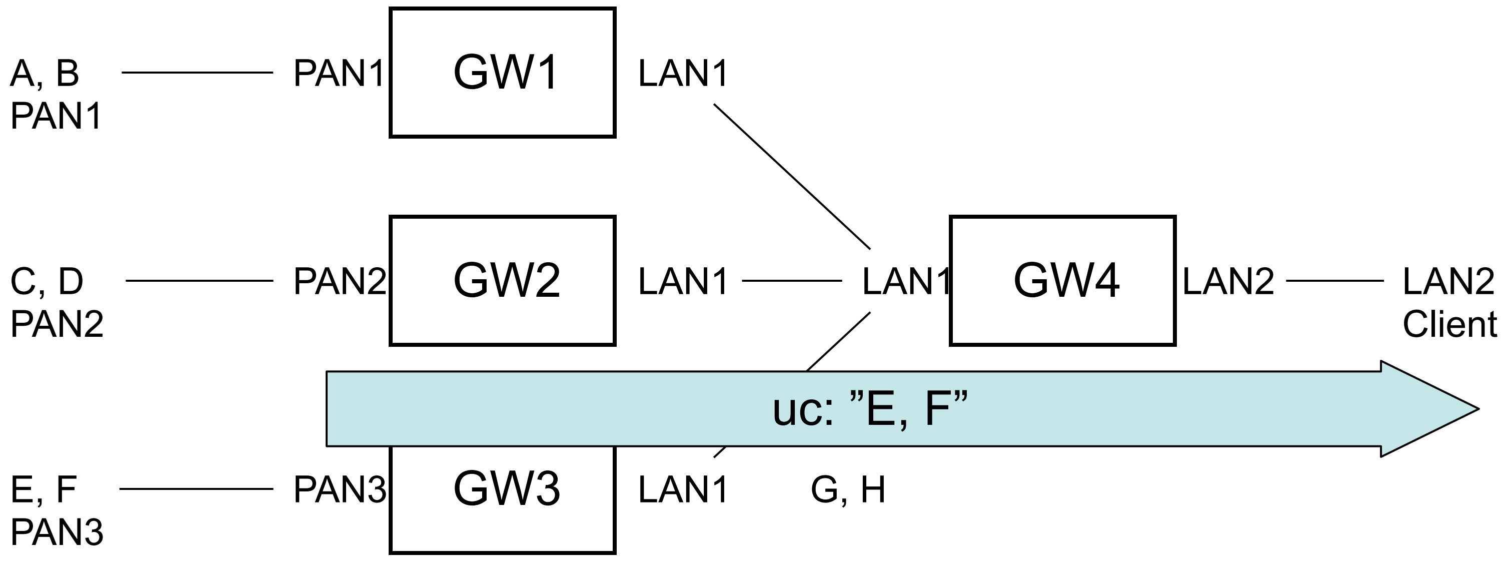
Example: Server Discovery



- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

- G
- H
- A
- B
- C
- D

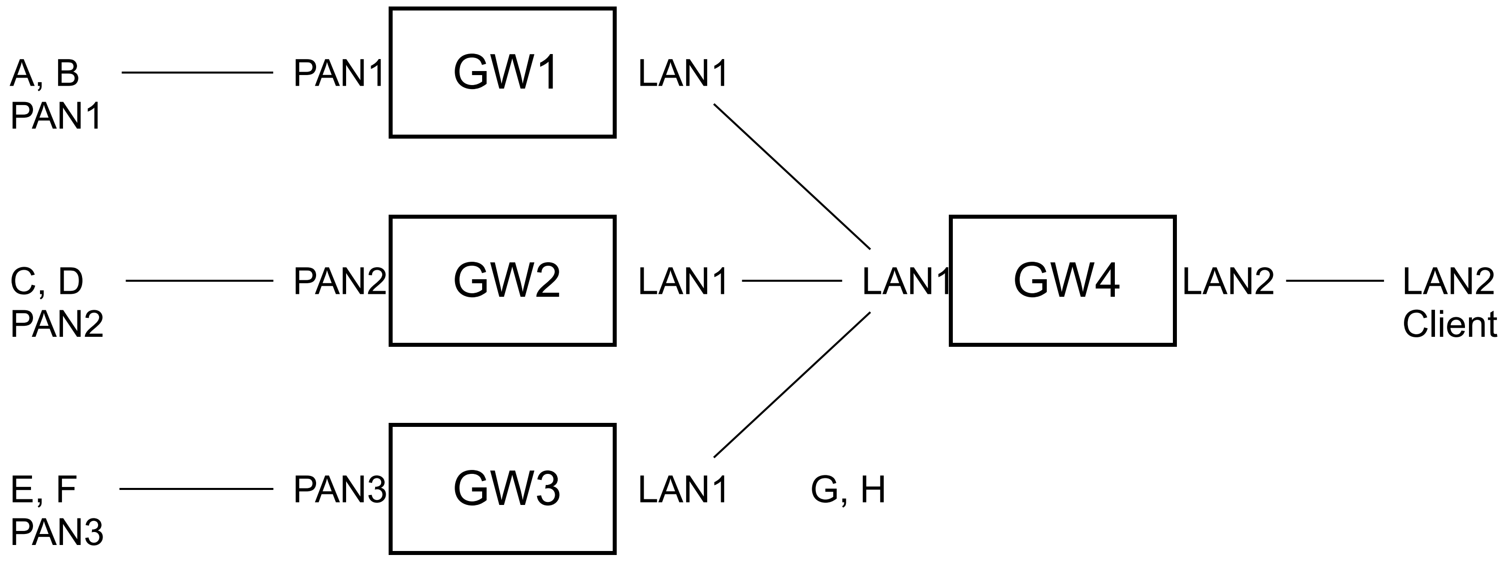
Example: Server Discovery



- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

- G
- H
- A
- B
- C
- D
- E
- F

Example: Server Discovery



- (on-link)
- GW4LAN1
- GW1PAN1
- GW2PAN2
- GW3PAN3

- G
- H
- A
- B
- C
- D
- E
- F

Further Work

Further Work

Further Work

- Advertizing node properties to caching discovery gateway

Further Work

- Advertizing node properties to caching discovery gateway
- M2M-style <u>SHORT</u>, well-defined paths for standard device types
  - Sensors
  - Dimmers
  - Generic On-Off devices
  - etc.

Further Work, Cont’d

Further Work, Cont'd

- Need for prefix NAT?
  - Attractive to keep LLN prefixes static
    - Sensor & Remote control associations to IP addresses are difficult to update
  - ISPs may assign dynamic IPv6 prefixes (?)
  - User may change ISP

Further Work, Cont'd

- Need for prefix NAT?
  - Attractive to keep LLN prefixes static
    - Sensor & Remote control associations to IP addresses are difficult to update
  - ISPs may assign dynamic IPv6 prefixes (?)
  - User may change ISP

- Coordination with other discovery drafts
  - What to add/change in this draft?
  - Inputs are welcome!

draft-brandt-coap-subnet-discovery

Anders Brandt
Sigma Designs
abr@sdesigns.dk

Thank You

`draft-brandt-coap-subnet-discovery`

Anders Brandt
Sigma Designs
abr@sdesigns.dk

80th IETF: core WG Agenda

13:00	Introduction, Agenda, Status	Chairs (10)
13:10	1 – link-format	ZS (5)
13:15	1 – core CoAP	ZS (25)
13:40	1 – block	CB (15)
13:55	1 – observe	KH (25)
14:20	2 – basic security	ZS (20)
14:40	discovery	CB+AB (20)
15:00 retire to Wednesday, 15:10	Intro	Chairs (05)
15:15	Group Communication	AR (10)
15:25	CoAP Usage	PV (10)
15:35	HTTP Mapping	KH+AC (15)
15:50	new: SOAP; WADL; overhead; security	GM ZS AC OG (5 each)
16:10 retire		

80th IETF: core WG Agenda

13:00	Introduction, Agenda, Status	Chairs (10)
13:10	1 – link-format	ZS (5)
13:15	1 – core CoAP	ZS (25)
13:40	1 – block	CB (15)
13:55	1 – observe	KH (25)
14:20	2 – basic security	ZS (20)
14:40	discovery	CB+AB (20)
15:00 retire to Wednesday, 15:10	Intro	Chairs (05)
15:15	Group Communication	AR (10)
15:25	CoAP Usage	PV (10)
15:35	HTTP Mapping	KH+AC (15)
15:50	new: SOAP; WADL; overhead; security	GM ZS AC OG
16:10 retire		(5 each)

Group Communication for CoAP

Akbar Rahman (Editor)
(with much input from Kerry Lynn, Peter Bigot,
Peter van der Stok, and others)

IETF 80, March 2011
http://tools.ietf.org/html/draft-rahman-core-groupcomm-04

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.

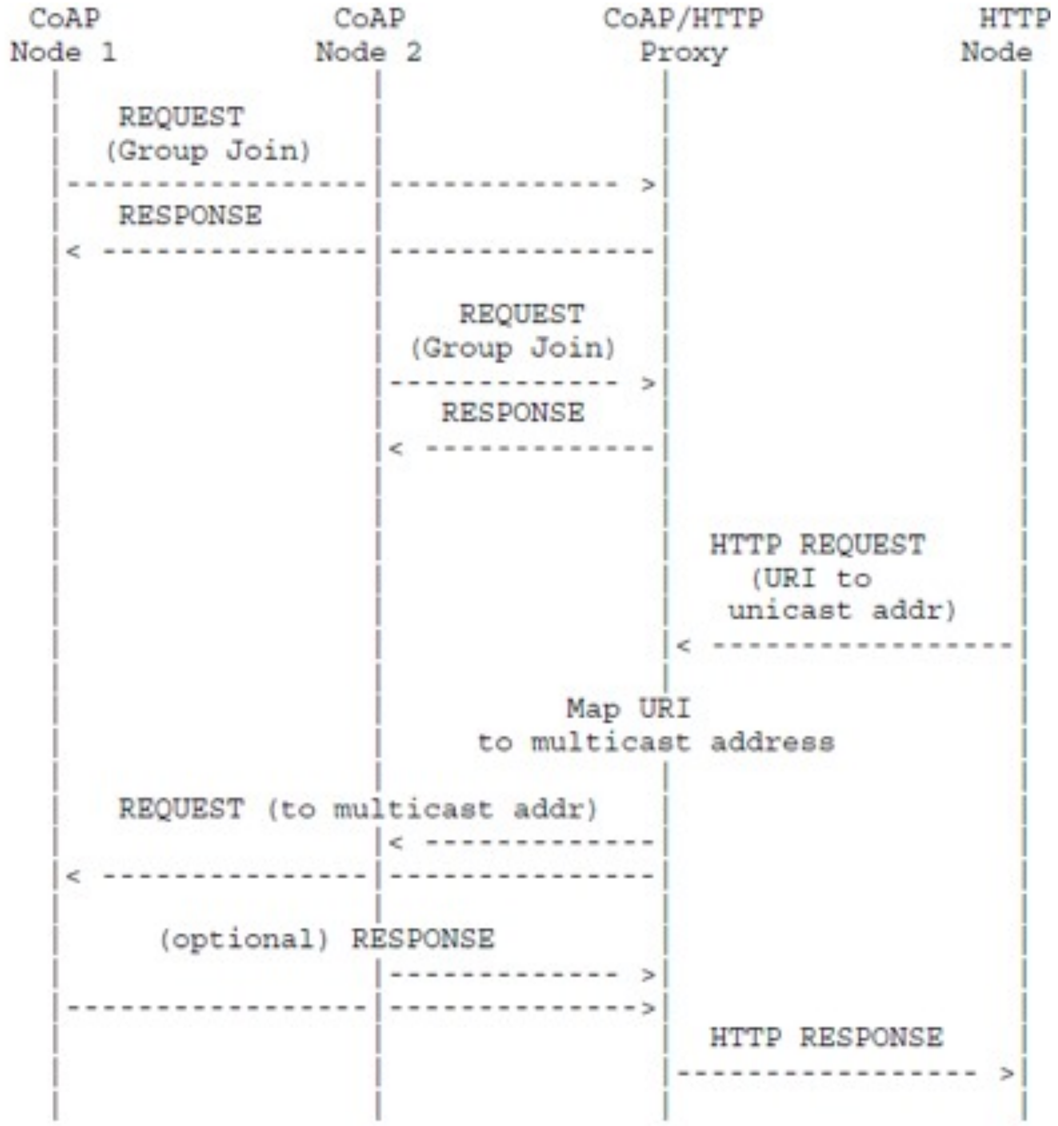
Group Security Requirements for CoAP (2/3)

- 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 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."

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)

Potential Approaches for Group Communication

- There are three alternative approaches for CoAP group communications each with associated pros/cons:
  - IP Multicast
  - Overlay (Proxy based) Multicast
  - CoAP Application level Group Management

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

80th IETF: core WG Agenda

13:00	Introduction, Agenda, Status	Chairs (10)
13:10	1 – link-format	ZS (5)
13:15	1 – core CoAP	ZS (25)
13:40	1 – block	CB (15)
13:55	1 – observe	KH (25)
14:20	2 – basic security	ZS (20)
14:40	discovery	CB+AB (20)
15:00 retire to Wednesday, 15:10	Intro	Chairs (05)
15:15	Group Communication	AR (10)
15:25	CoAP Usage	PV (10)
15:35	HTTP Mapping	KH+AC (15)
15:50	new: SOAP; WADL; overhead; security	GM ZS AC OG
16:10 retire		(5 each)

CoAP Utilization for Building Control

draft-vanderstok-core-bc-03

Grouping/Discovery/Legacy/Commissioning

Peter van der Stok;
Kerry Lynn

March 30, 2011

Groups and multicast

A typical BC installation may have 1000s of “points”
Group members are typically within 1-2 hop distance from source

Authority:

Node (host [:socket]) resolves to a unicast IP address
Group (set of nodes) resolves to a scoped multicast group IP address or set of serial IP unicasts (ref: group-comm I-D)

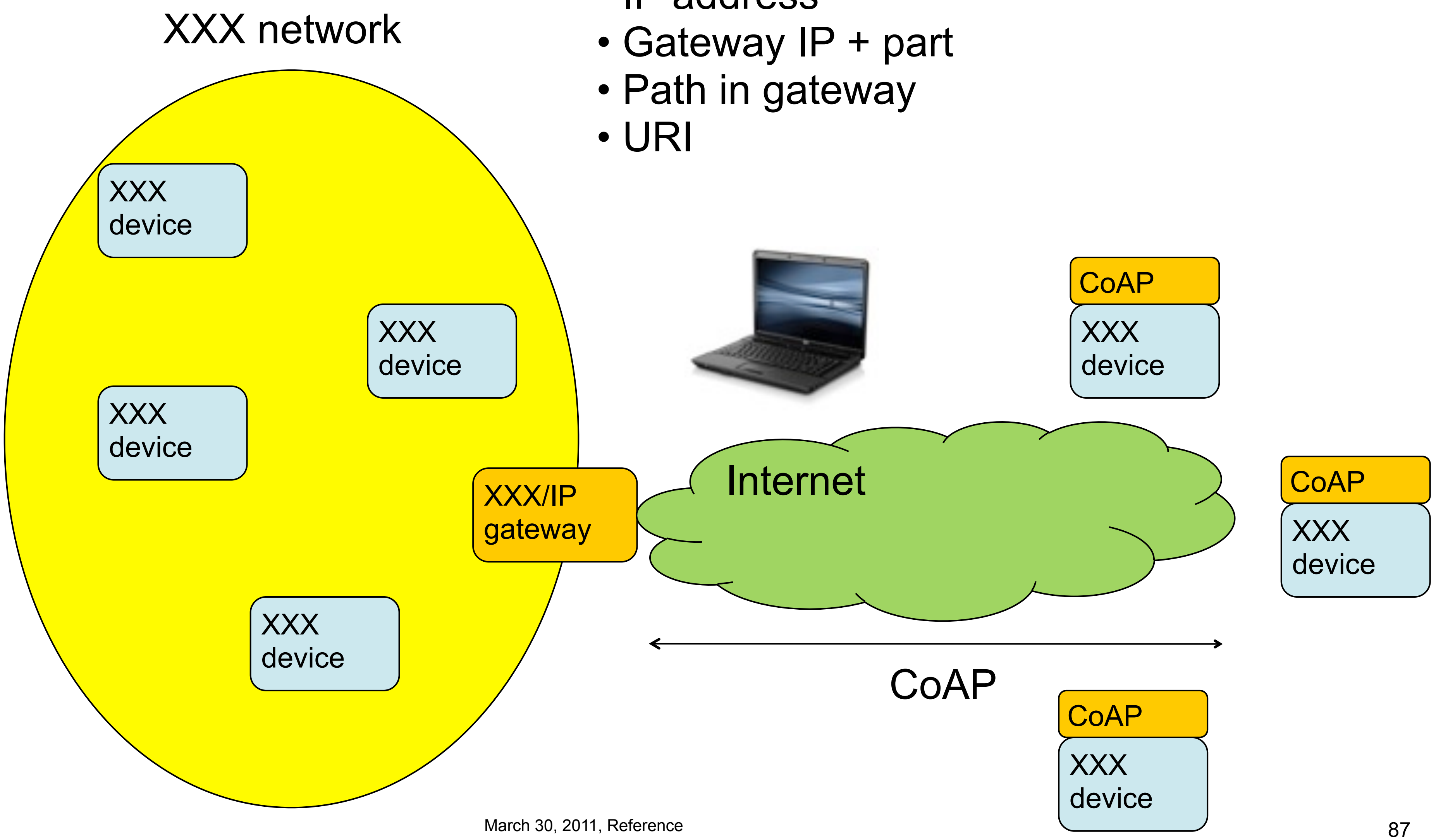
DNS stores multicast group IP address in A or AAAA record

Group operation is sent to multicast group IP address.
The resource is identified with a single path on all group members

Assuming hosts with legacy representations (ZigBee, BACnet, ..)
End-point of a given legacy XXX function is prefixed by:
/.well-known/XXX (we refer to this as the “schema”)

XXX legacy gateway

XXX host behind gateway represented by:
- IP address
- Gateway IP + part
- Path in gateway
- URI



core-bc Describes DNS-based Service Discovery

Central server solves:

Large set > 100 nodes per domain; Grouping (over subnets)

DNS-SD:

Based on mature, well-known technology
Service instance name is of form
<Instance>.<ServiceType>.<Location>
Subtypes: e.g. _zigbee._sub._coap._udp
These labels used to create PTR and SRV records

TXT records contain “key=value” attributes
Proposed attribute:
schema=.well-known/XXX (XXX legacy representation)

Proposed Extension to link-format to Support DNS-SD

Each link that defines a functional entry point should provide the following attributes:

- sn = service instance name (unique within domain)
- st = service type (defaults to “coap”)
- ss = service subtype

Read by an installation tool and used to create DNS-SD records.

Satisfies REQ8 of coap-req

Location: lamp4/office5/hilton8.org

sn: 545aafgh678uu8
ss: _light._zigbee
IP: ff02::45

Barcode readable

IP: fdfd::1234

DNS-SD
identifier: 545aafgh678uu8._light._zigbee._sub._coap._udp.hilton8.

AAAA:	fdfd::1234
SRV:	lamp4/office5/hilton8.org; portyy
TXT:	schema=zigbee ss=_light._zigbee

Example of commissioning procedure

Assume an installation tool, DNS server on-line
DNS is initialized with domain(s)
Host has link-local address plus possibly others

Need <Instance>.<ServiceType>.<Location> triple to create DNS-SD records

Location = DNS domain name (*zone)*
Instance = based on function + EUI-64 or vendor s/n;
    readable by bar coder of tool
    buffered, plus injected into device as "sn" attribute
ServiceType = read from device as "st" attribute
Service Subtype = read from device as "ss" attribute

Installation tool collects attributes from device (including key=value) and installs PTR, SRV, AAAA and TXT records into DNS server

80th IETF: core WG Agenda

13:00	Introduction, Agenda, Status	Chairs (10)
13:10	1 – link-format	ZS (5)
13:15	1 – core CoAP	ZS (25)
13:40	1 – block	CB (15)
13:55	1 – observe	KH (25)
14:20	2 – basic security	ZS (20)
14:40	discovery	CB+AB (20)
15:00	retire to Wednesday, 15:10 Intro	Chairs (05)
15:15	Group Communication	AR (10)
15:25	CoAP Usage	PV (10)
15:35	HTTP Mapping	KH+AC (15)
15:50	new: SOAP; WADL; overhead; security	GM ZS AC OG
16:10	retire	(5 each)

HTTP mapping

- **Charter item:**

  There also may be proxies that interconnect between other Internet protocols and the Devices using the CoAP protocol. The WG will define a mapping from CoAP to an HTTP **REST API**; this mapping will not depend on a specific application.

- **Not quite clear which direction is meant (which end is the server and which end is the client).**
- **Whatever we define will be generic, i.e. not for a specific application or non-REST use of HTTP.**

HTTP Mapping

Klaus Hartke

HTTP Mapping

WG is chartered to define a mapping between **CoAP** and **HTTP**

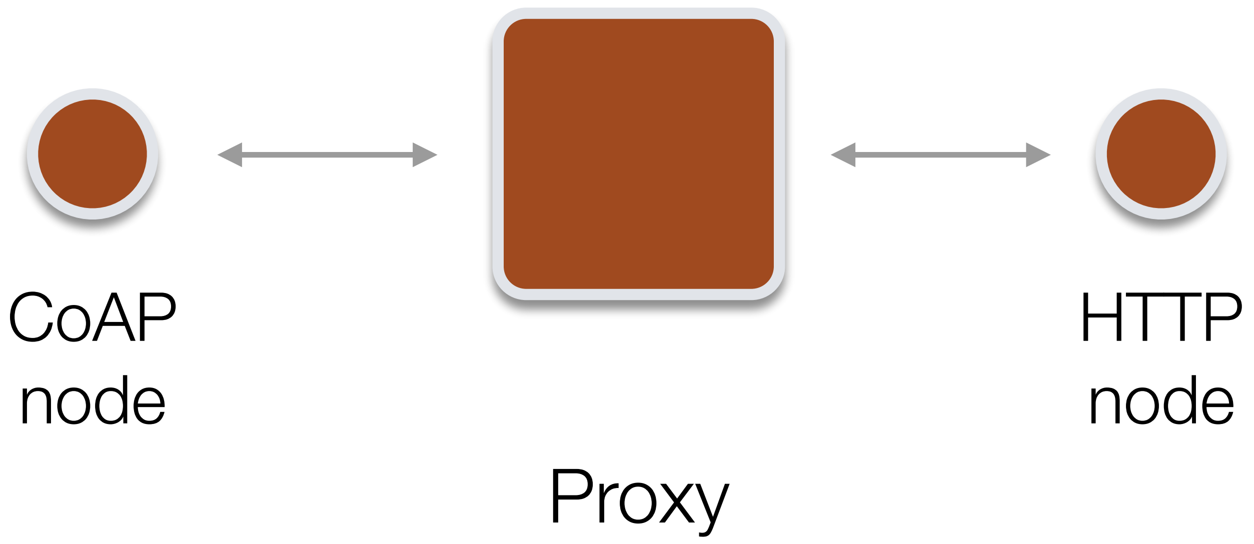
Need to figure out

1. how it works
2. what to put in the draft (i.e. what needs to be standardised so implementations will interoperate)

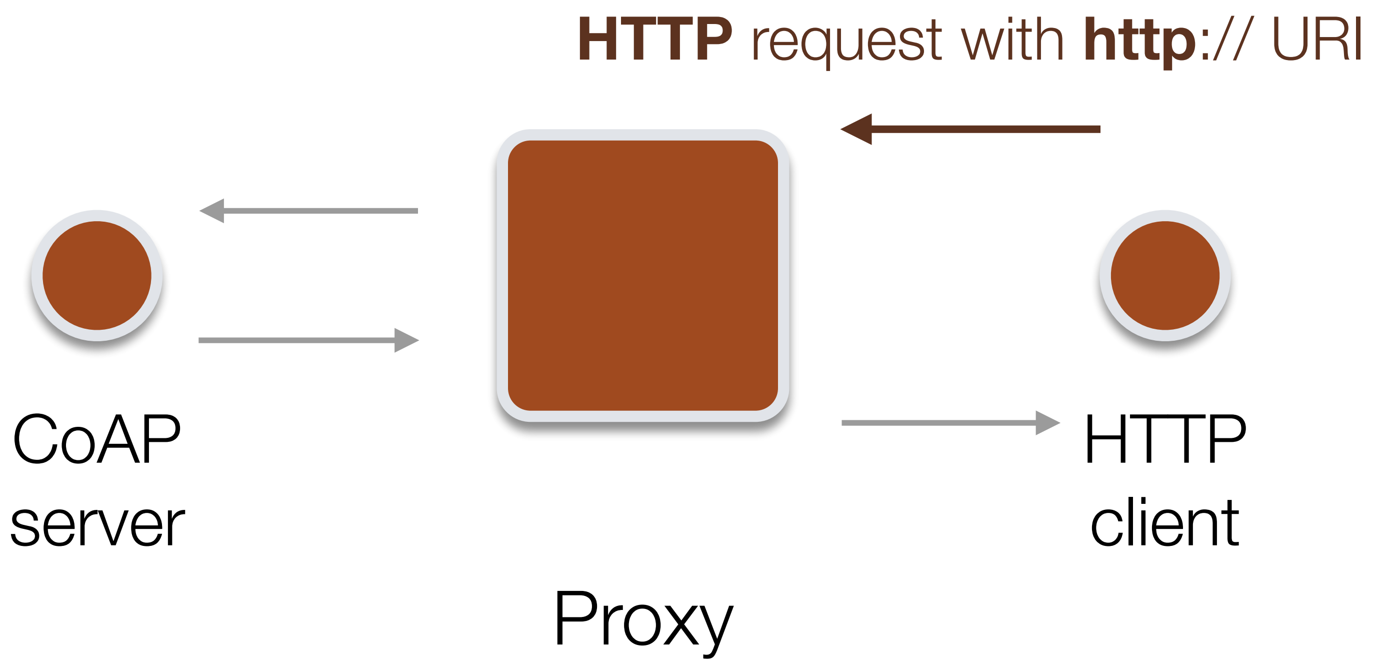
The current specification goes into more detail than is needed

Proposal: clear separation between the specification and possible implementations

CoAP node
HTTP node

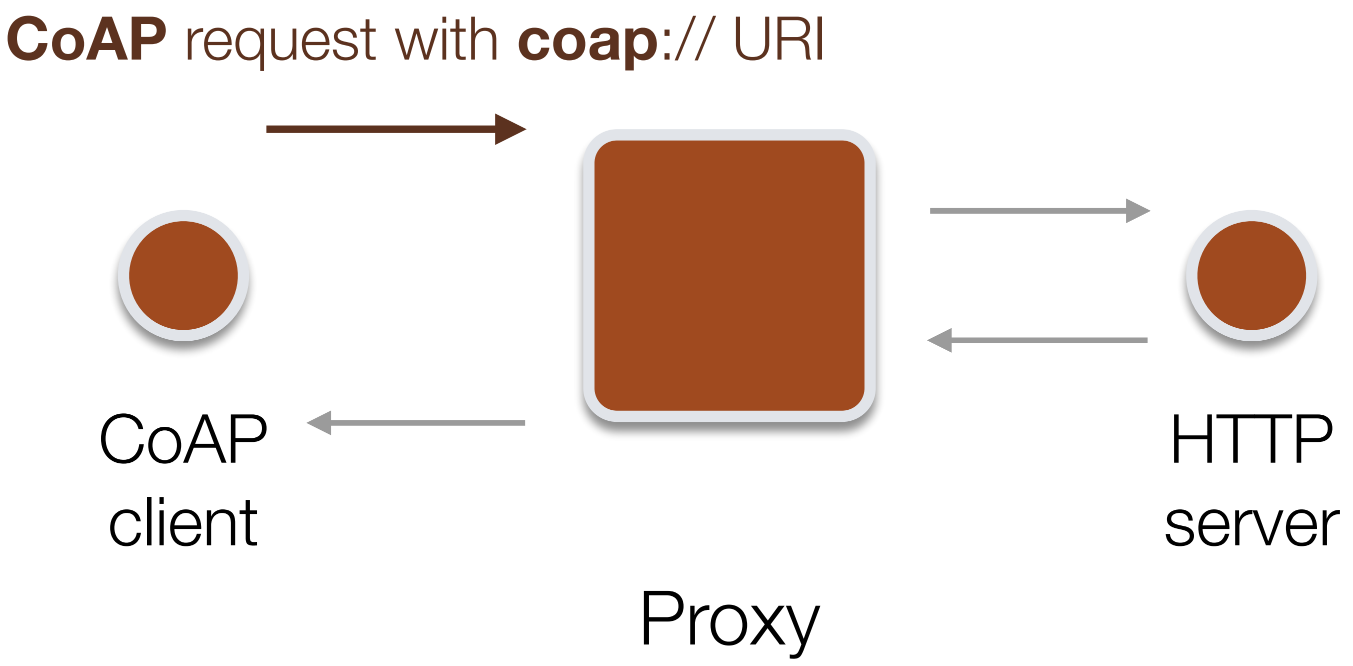

CoAP node
Proxy
HTTP node

HTTP/CoAP Reverse Proxy



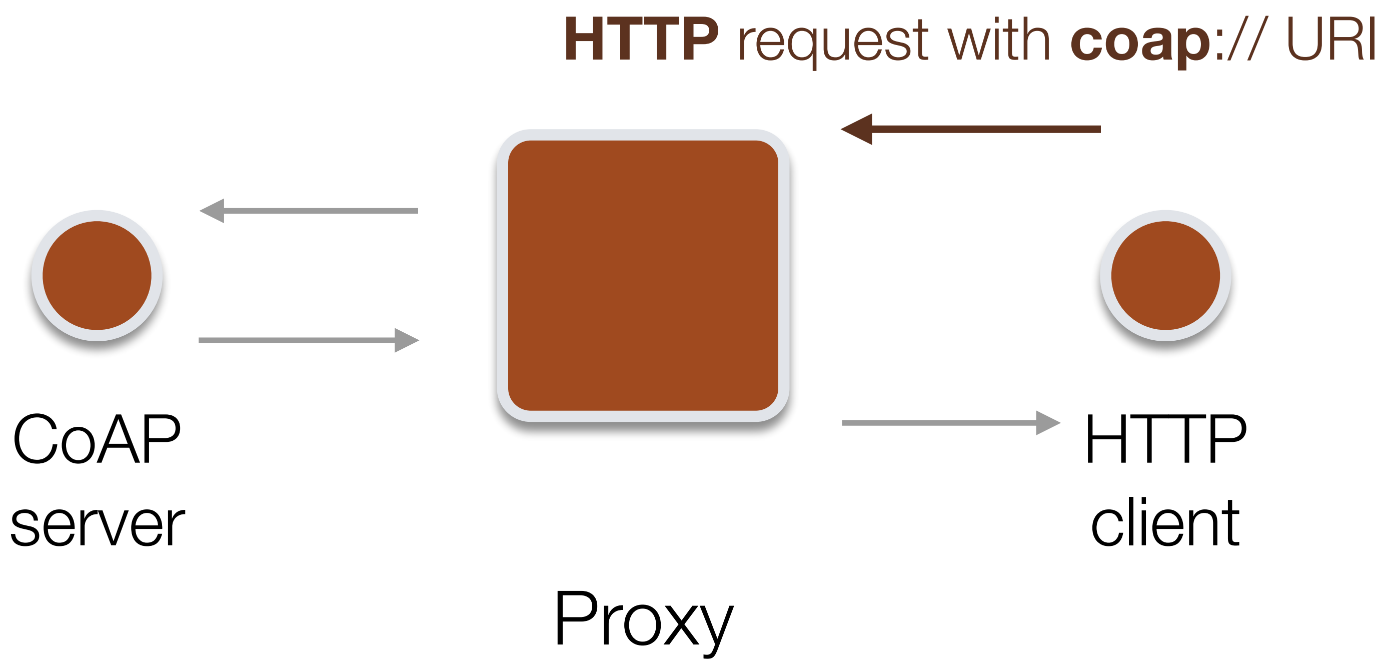
- proxy appears to the client just like an ordinary HTTP server
- no special configuration on the HTTP client is necessary
- proxy decides where to send CoAP requests

CoAP/HTTP Reverse Proxy



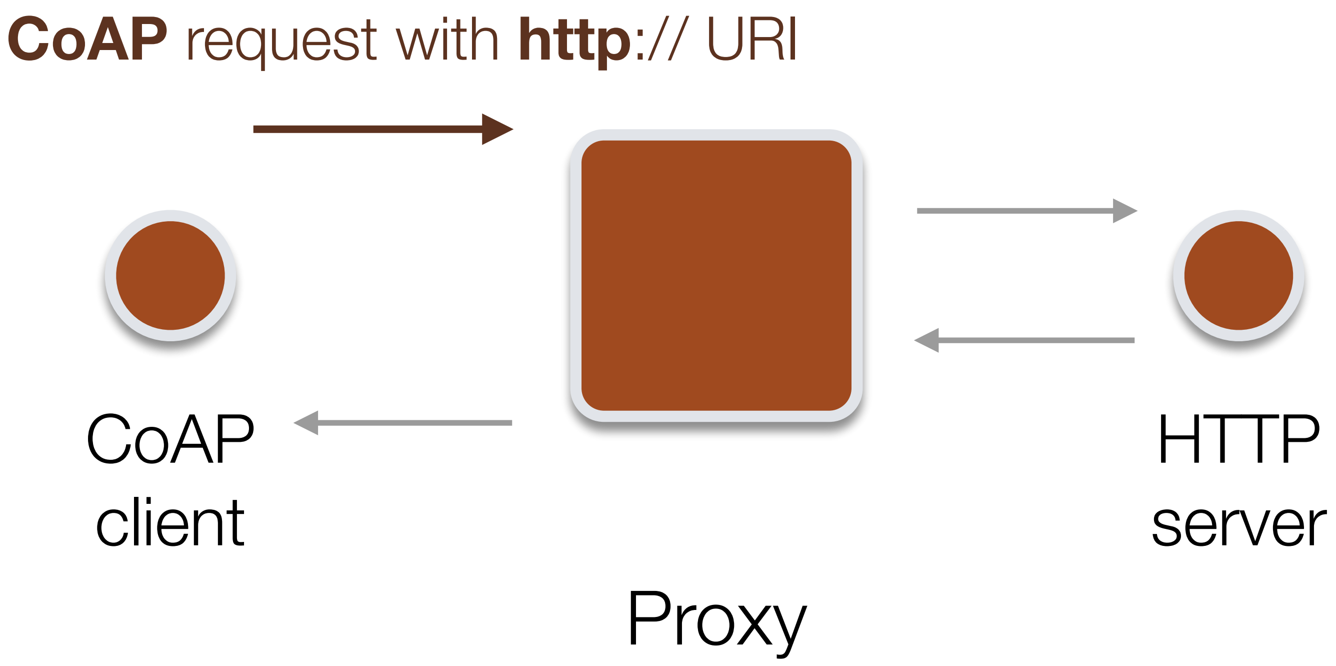
- proxy appears to the client just like an ordinary CoAP server
- no special configuration on the CoAP client is necessary
- proxy decides where to send HTTP requests

HTTP/CoAP Forward Proxy



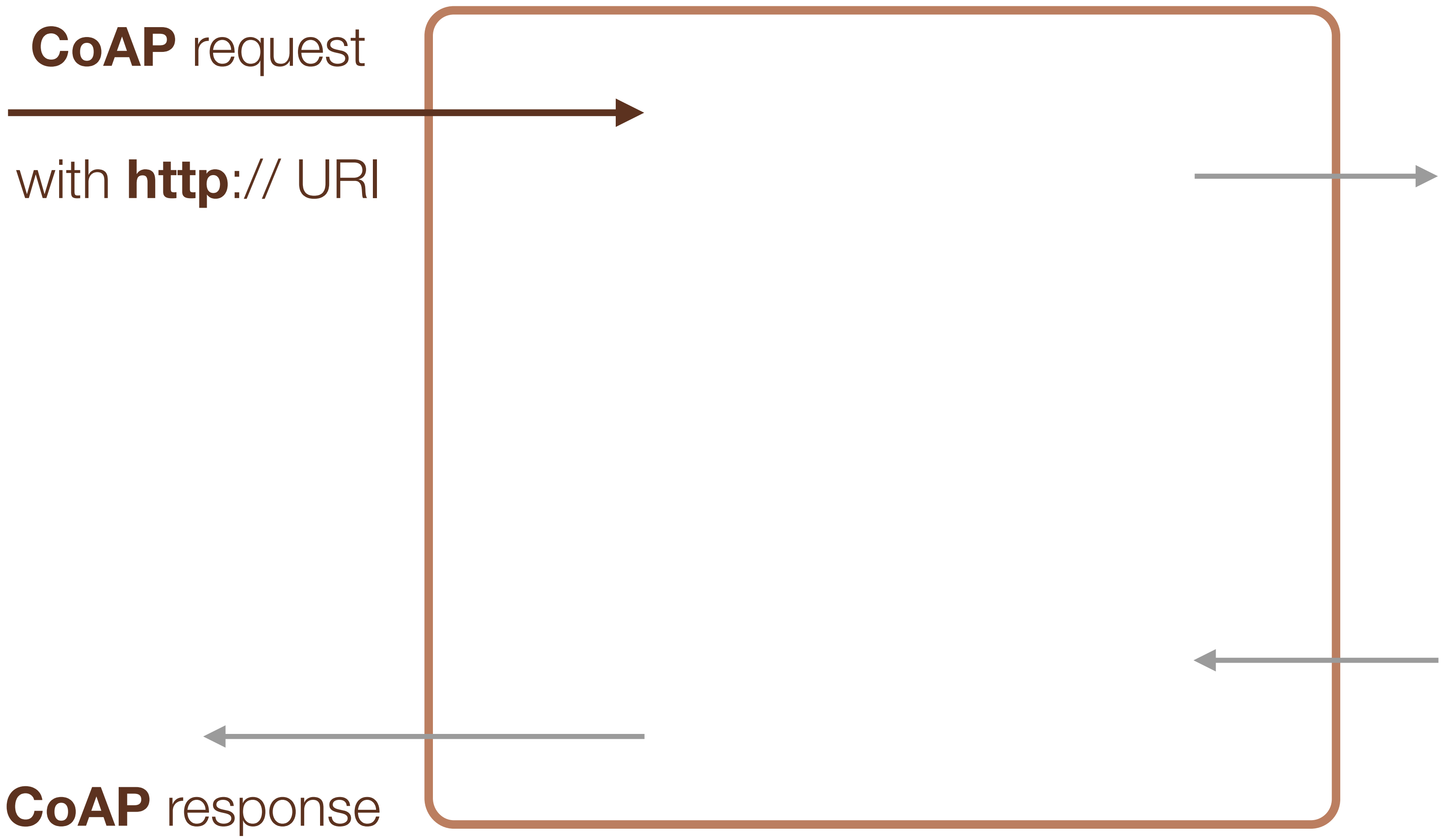
- client sends HTTP request naming the CoAP server as target
- client must be specially configured to use the proxy
- proxy sends CoAP requests to the CoAP server

CoAP/HTTP Forward Proxy

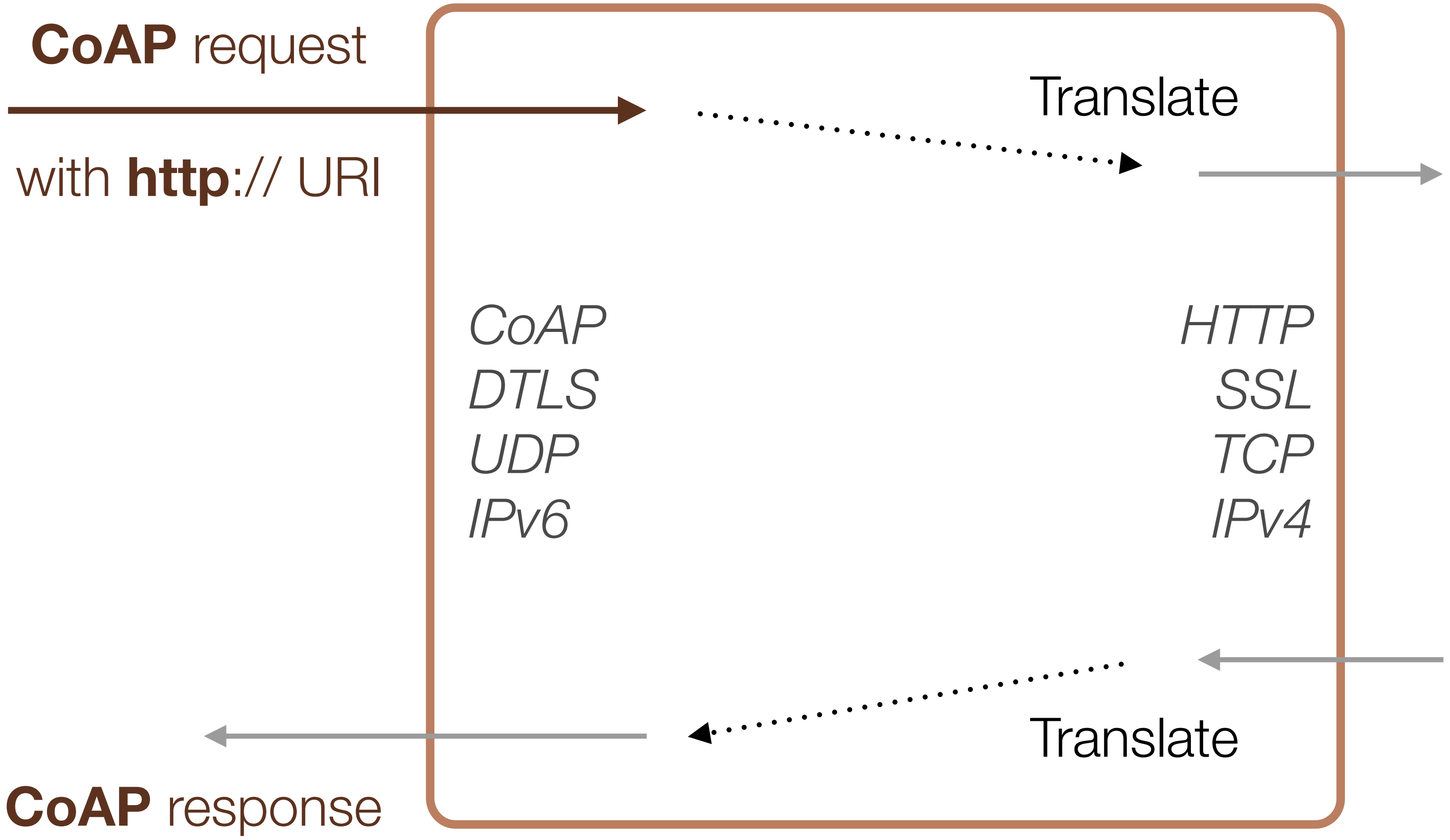


- client sends CoAP request naming the HTTP server as target
- client must be specially configured to use the proxy
- proxy sends HTTP requests to the HTTP server

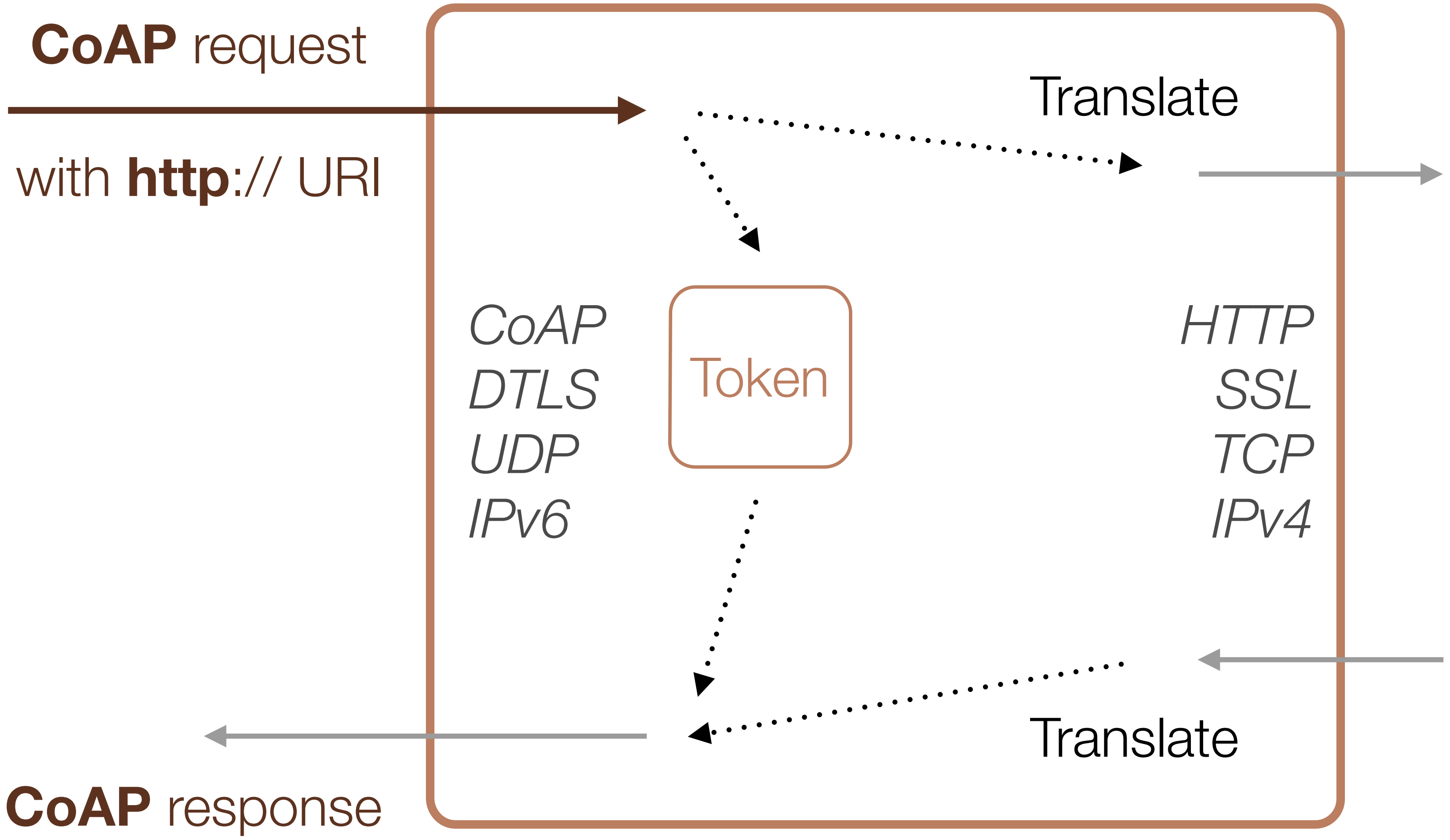
Proxy Implementation



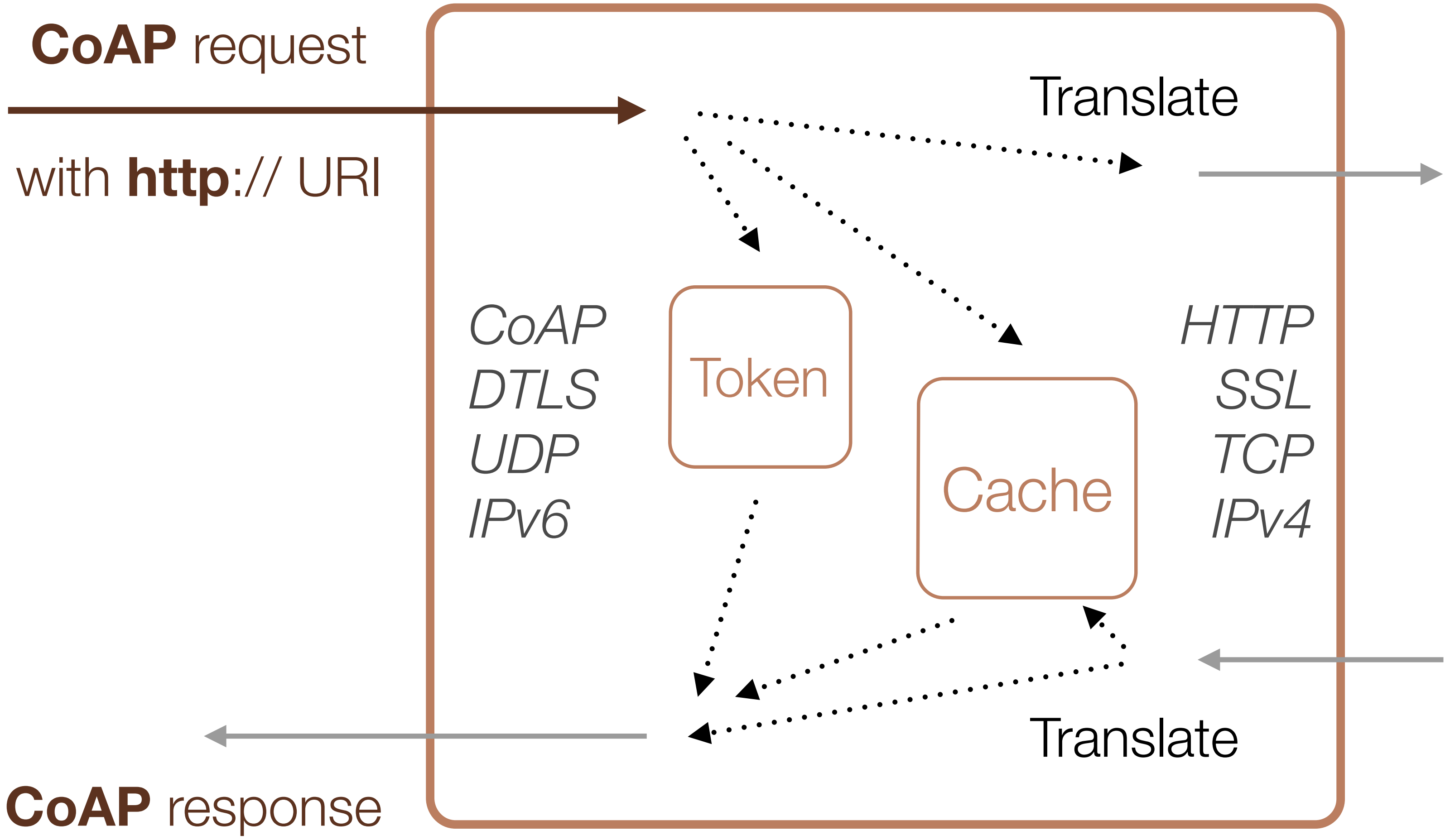
Proxy Implementation



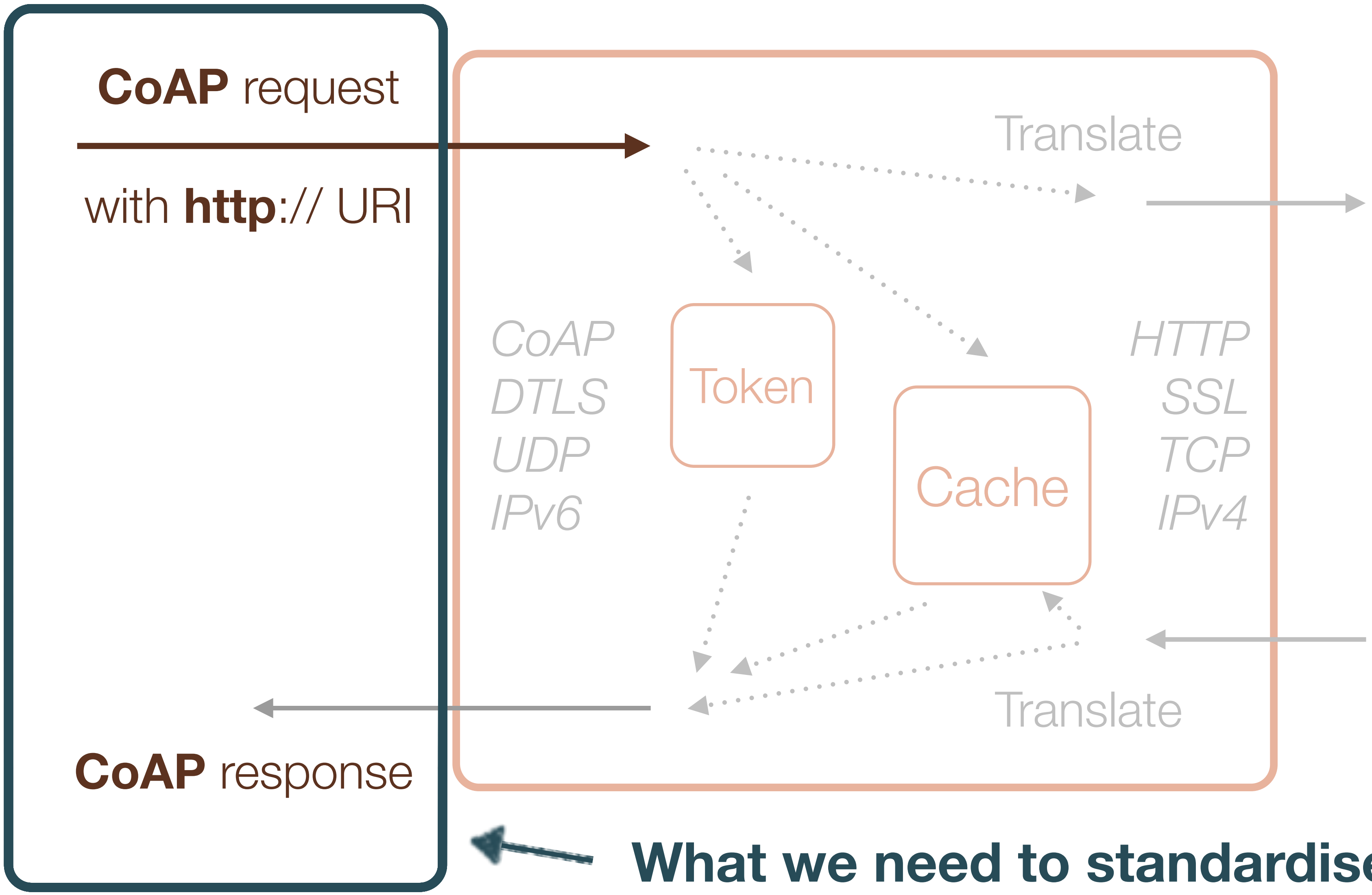
Proxy Implementation



Proxy Implementation



Proxy Implementation



What we need to standardise

Specification

Define for any

HTTP request with an **http**:// URI

CoAP request with a **coap**:// URI

CoAP request with an **http**:// URI

HTTP request with a **coap**:// URI

the response that the proxy is expected
to return to the client

Specification

Define for any

HTTP request with an **http**:// URI

RFC 2616

CoAP request with a **coap**:// URI

draft-ietf-core-coap

CoAP request with an **http**:// URI

draft-hartke-core-coap-http

HTTP request with a **coap**:// URI

to do

the response that the proxy is expected
to return to the client

draft-hartke-core-coap-http-00

Section 2 **Specification**

Defines for any
CoAP request with an **http**:// URI
the response that the proxy is expected to return to the client

Section 3 **Implementation considerations**

Payload conversion, ETag mapping, Block-wise transfers, Redirect handling, Examples, ...

draft-castellani-http-coap-mapping-01

A reference for HTTP-CoAP mapping implementors

Angelo P. Castellani and Salvatore Loreto

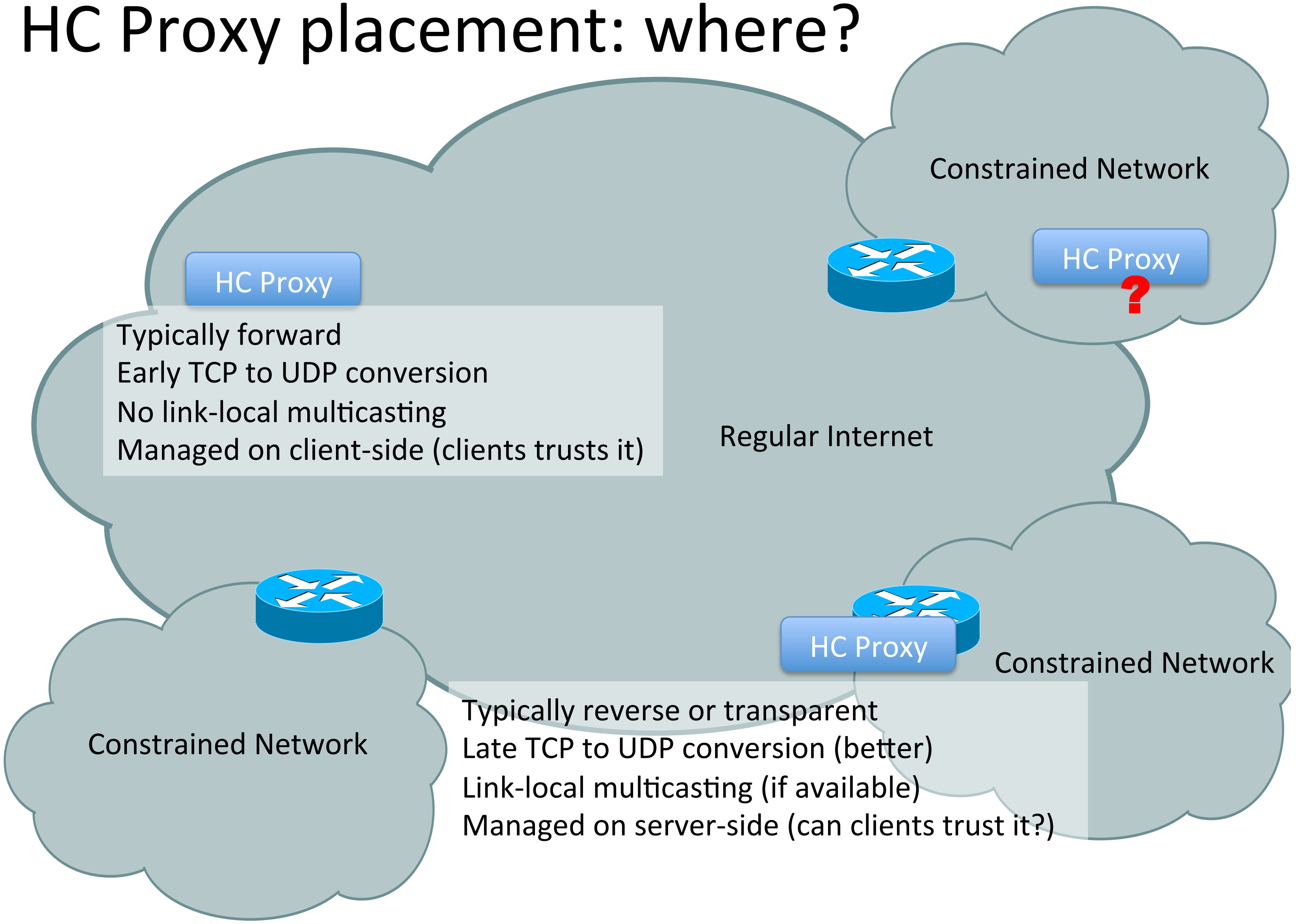
Proxy Terminology

- HTTP-CoAP forward proxy (client knows it)
  - Client knowledge required
- HTTP-CoAP reverse proxy (server is known by it)
  - Proxy knowledge required
- HTTP-CoAP transparent proxy (aka intercepting)
  - No knowledge required at all (+1)

HTTP mapping: current issues

- Where the CoAP proxy should be placed?
- Should we handle HTTP on IPv4 and CoAP on IPv6?
- Is DNS useful to perform HTTP-CoAP mapping?
- Do we want transparent URI mapping?
- How do we map CoAP multicast to HTTP unicast?
- How do we map CoAP observe directly to HTTP?

HC Proxy placement: where?



Typically forward
Early TCP to UDP conversion
No link-local multicasting
Managed on client-side (clients trusts it)

Typically reverse or transparent
Late TCP to UDP conversion (better)
Link-local multicasting (if available)
Managed on server-side (can clients trust it?)

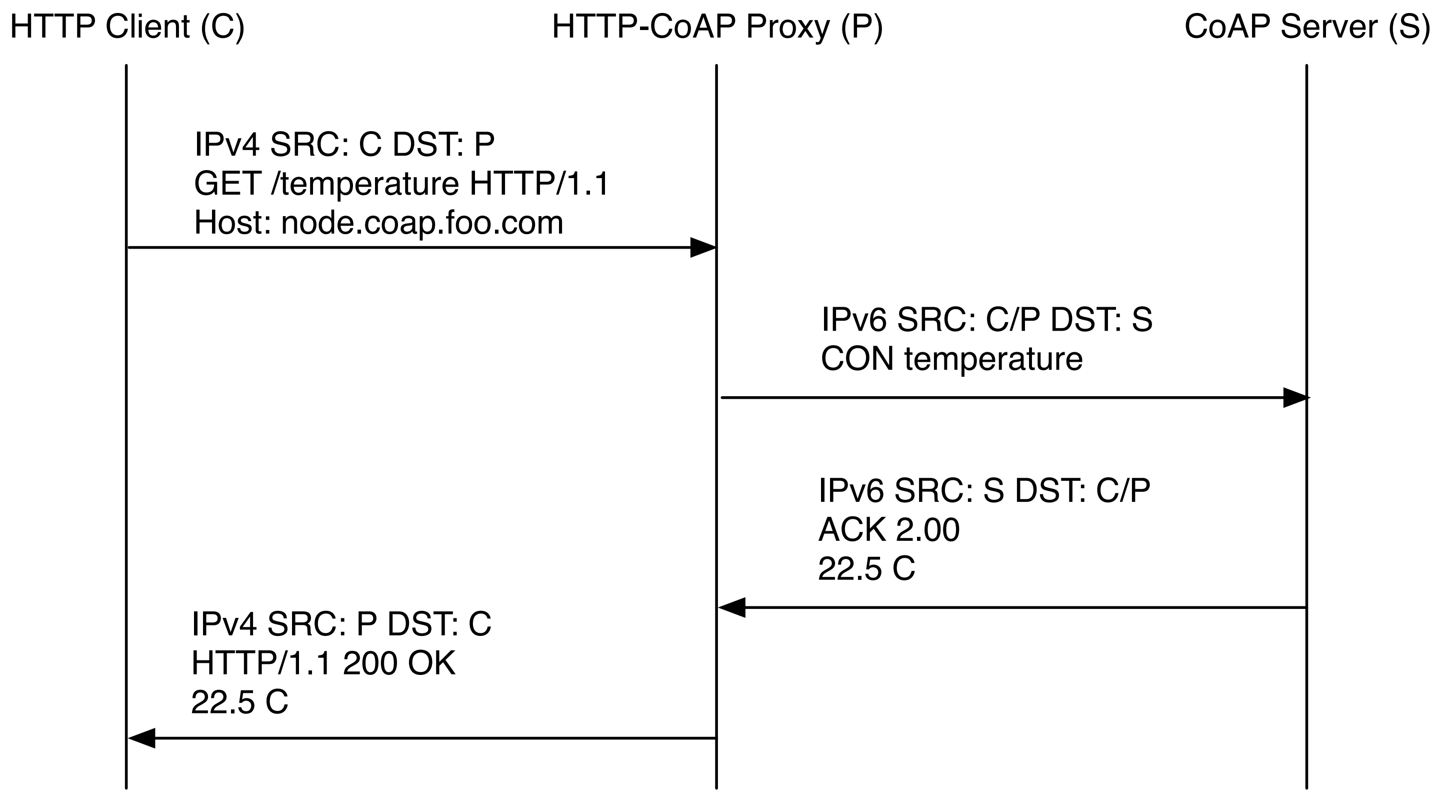
HTTP/IPv4 to CoAP/IPv6 using DNS

- IAB workshop result: IP version mismatch?
  - Typically HTTP clients run over IPv4
  - CoAP nodes are expected to use IPv6/6LoWPAN
- A simple solution: DNS
  - node.coap.foo.bar A record points to the proxy
  - node.coap.foo.bar AAAA record points to the node
- HTTP/IPv4 clients will be directed to the proxy
- Proxy identifies the node w/ HTTP/1.1 Host header
  - Can also identify groups of nodes (e.g. multicast addr)

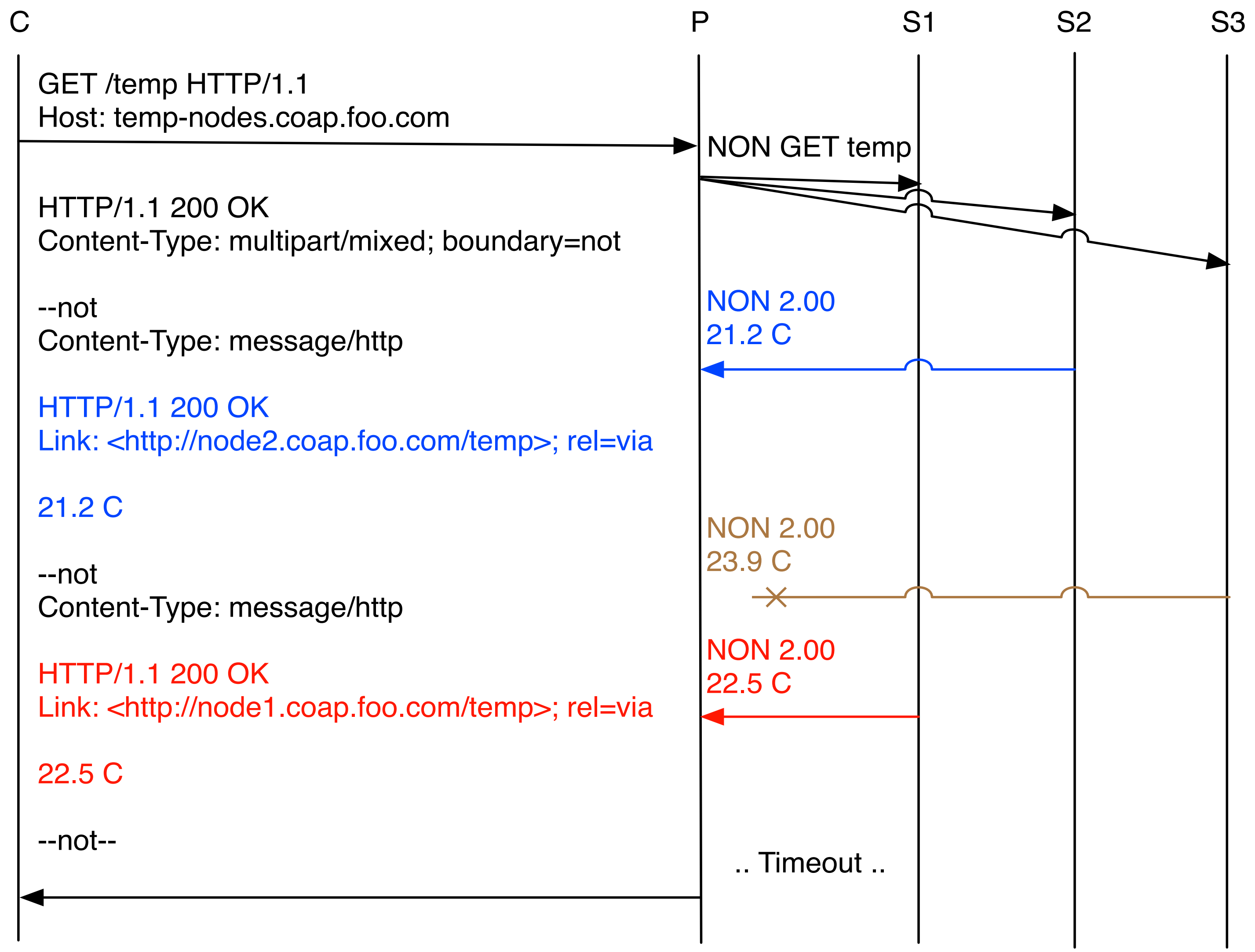
HTTP-CoAP: transparent URI mapping

- coap:// resources accessible using HTTP
  - CoAP server offers coap://node.coap.foo.com/temperature
  - HTTP clients access it at http://node.coap.foo.com/temperature
  - The URI does not change! (except the schema)

HTTP/IPv4-CoAP/IPv6 Example



HTTP-CoAP: unicast to multicast



```
GET /temp HTTP/1.1
Host: temp-nodes.coap.foo.com

HTTP/1.1 200 OK
Content-Type: multipart/mixed; boundary=not

--not
Content-Type: message/http

HTTP/1.1 200 OK
Link: <http://node2.coap.foo.com/temp>; rel=via

21.2 C

--not
Content-Type: message/http

HTTP/1.1 200 OK
Link: <http://node1.coap.foo.com/temp>; rel=via

22.5 C

--not--
```

80th IETF: core WG Agenda

13:00	Introduction, Agenda, Status	Chairs (10)
13:10	1 – link-format	ZS (5)
13:15	1 – core CoAP	ZS (25)
13:40	1 – block	CB (15)
13:55	1 – observe	KH (25)
14:20	2 – basic security	ZS (20)
14:40	discovery	CB+AB (20)
15:00	retire to Wednesday, 15:10 Intro	Chairs (05)
15:15	Group Communication	AR (10)
15:25	CoAP Usage	PV (10)
15:35	HTTP Mapping	KH+AC (15)
15:50	new: SOAP; WADL; overhead; security	GM ZS AC OG
16:10	retire	(5 each)

SOAP over CoAP - Motivation

- SOAP is basis for Devices Profile for Web Service (DPWS)
  - SOAP profile for ‚constrained‘ devices including Discovery, Eventing and Security
  - Aligned with and extendable by W3C SOAP Web services framework
  - Seamless integration of devices in business infrastructures
- SOAP (in constrained environments) transport via
  - SOAP over HTTP: suffering from TCP
  - SOAP over UDP: suffering from unreliability
  - <u>SOAP over CoAP: </u>lightweight and reliable
- EXI SOAP over CoAP is mapping seamless with XML SOAP over HTTP

SOAP over CoAP

- Map DPWS (not SOAP!) one- and two-way exchange patterns on CoAP message layer
- Use POST method and appropriate response codes
- Supported media type XML Infoset (SOAP+XML, EXI, FI)
- Response destination addressing (i.e. WS-Addressing anonymous endpoint)
- Map WS-Addressing message id on CoAP token option

Foo over CoAP - Issues

- draft-ietf-core-block-02
  - Payload only for either request or response ✓
- draft-ietf-core-coap-05
  - POST has no payload in response ✓
  - User defined header options (i.e. SOAPAction)
    - For SOAP (not DPWS!) one-way MEP
    - Use unassigned header options

Advertisement block: draft-bormann-coap-misc-07

- **User-defined option:**
  - **reuse option number 14 (first fencepost)**
  - **elective**
  - **any length > 0 is a user-defined option**
  - **recommended to start with SDNV of enterprise number**
    - static byte string -- makes it easy to allocate a unique value

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 0 0 0 0 0 0 1|0 1 1 1 0 0 1 1| private opt-id|   value...    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
\___SDNV of enterprise number___/

   Figure 1: Example option value for user-defined option
```

draft-castellani-coap-overhead-01

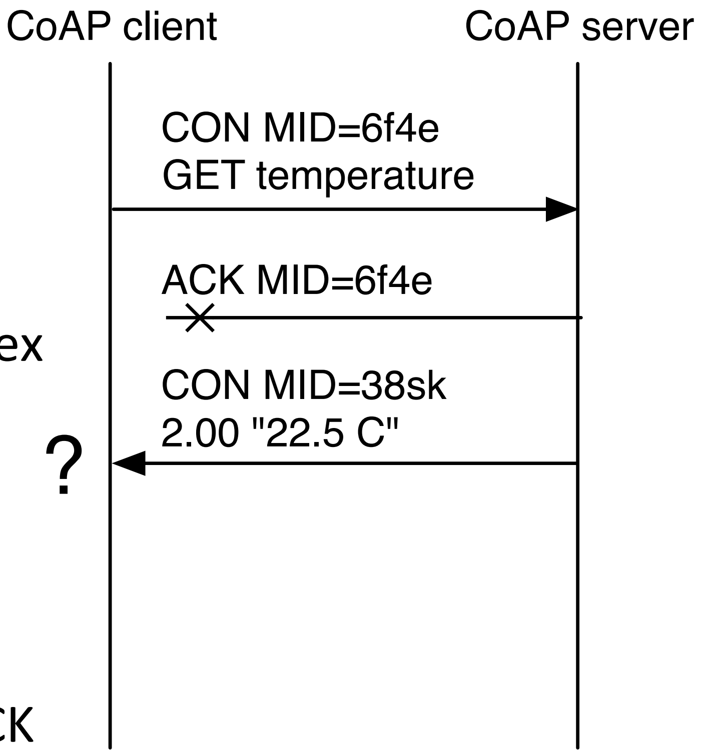
Angelo P. Castellani and Mattia Gheda

Avoid Token to reduce complexity

- With Token
  - Simple server devices allocate 8B more per request
  - Adds 9B overhead to non-empty packets
  - Session matching is hard
    - CON/NON pkts: match (IP, port, Token)
    - Non-empty ACK pkts: match (IP, port, Token, MID)
    - Empty ACK/RST pkts: match (IP, port, MID)
- No Token
  - Less overhead and memory consumption
  - Simpler session matching:
    - CON/NON: match (IP, port)
    - ACK/RST: match (IP, port, MID)

New virtual message type: CONACKs

- What if the ACK is lost?
- Current:
  - If no ACK is received the CON MUST be retransmitted
  - Not efficient and more complex to implement!
- Proposed
  - New virtual message type: CONACKs
  - A new CON piggybacks the ACK



New header proposal

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver| T |  OC   |M|  A  |  AC   |      SMID     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- Action (A): identifies empty msgs, requests, 1xx, 2xx, 3xx, 4xx, 5xx, 6xx responses
- Action Code (AC): gives details about the specific request (GET, POST, etc.) or response
- Sequential MID (SMID): sequentially growing message ID.
  - Do we need more than 8 bits?
- More parts (M): message is not complete. (optional)

Example: bidirectional large messages

```
Client                                               Server
  |                                                     |
  | CON [M=1,SMID=100], POST, /configuration --->|
  |                                                     |
  |<------------------------- ACK [M=0,SMID=100] |
  |                                                     |   CONACK
  | CON [M=0,SMID=101] ------------------------->|
  |                                                     |
  |<---------------- CON [M=1,SMID=102], 2.00 OK |
  |                                                     |
  | ACK [M=0,SMID=102] ------------------------->|
  |                                                     |
  |<------------------------- CON [M=0,SMID=103] |
  |                                                     |
  | ACK [M=0,SMID=103] ------------------------->|
  |                                                     |
```

Security: What does the charter say?

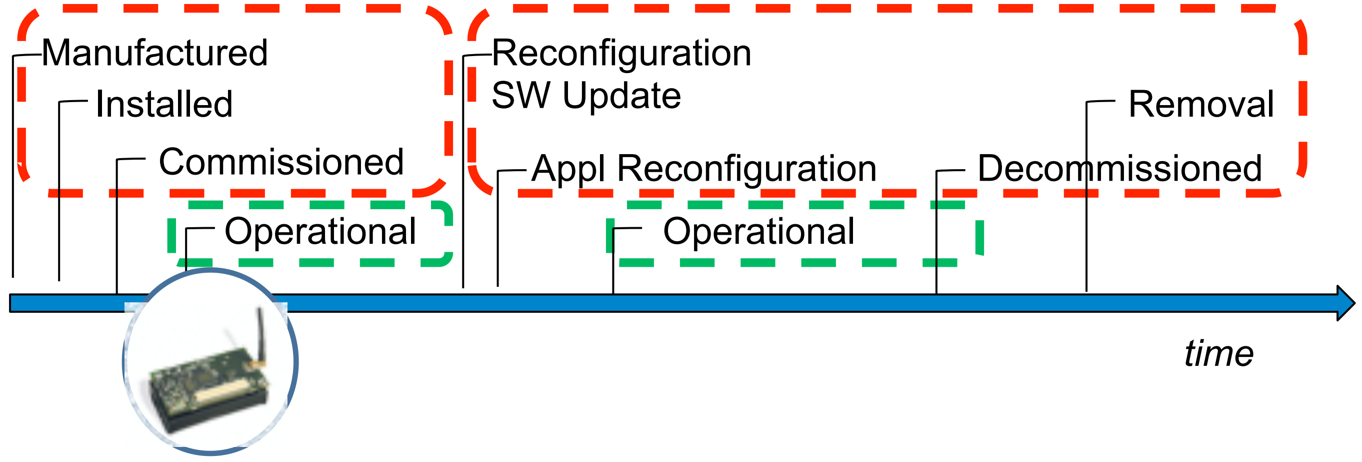
- Security, particularly keying of new Devices, is very challenging for these applications.
- The WG will work to select approaches to security bootstrapping which are realistic given the constraints and requirements of the network.
- To ensure that any two nodes can join together, all nodes must implement at least one universal bootstrapping method.

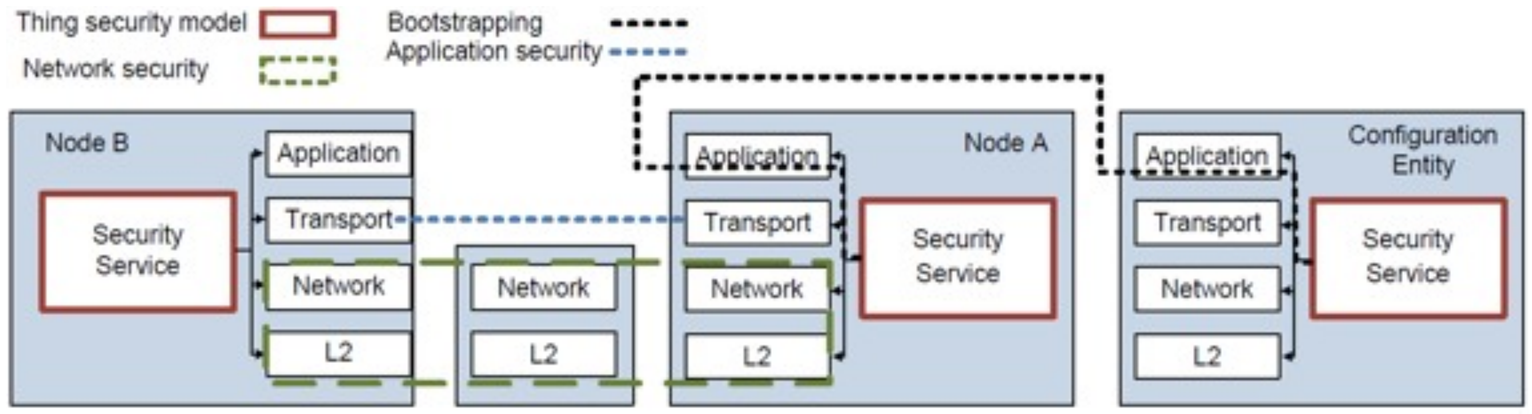
Security Considerations in the IP-based Internet of Things

Oscar Garcia-Morchon,
Sye Loong Keoh, Sandeep S. Kumar, René Hummen and *Rene Struik*

Philips Research, RWTH Aachen, *Struik Security Consultancy*

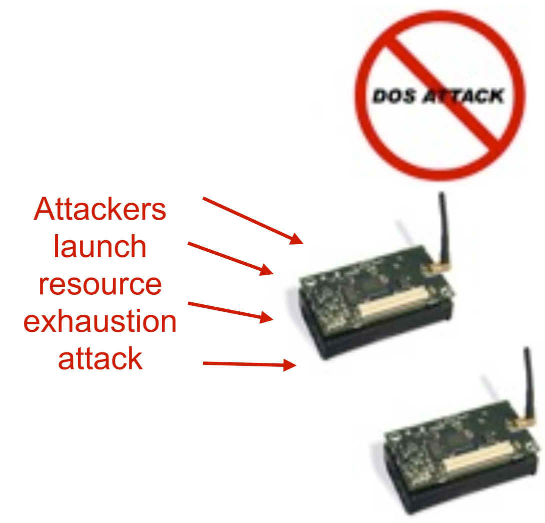
Thing lifecycle and security framework

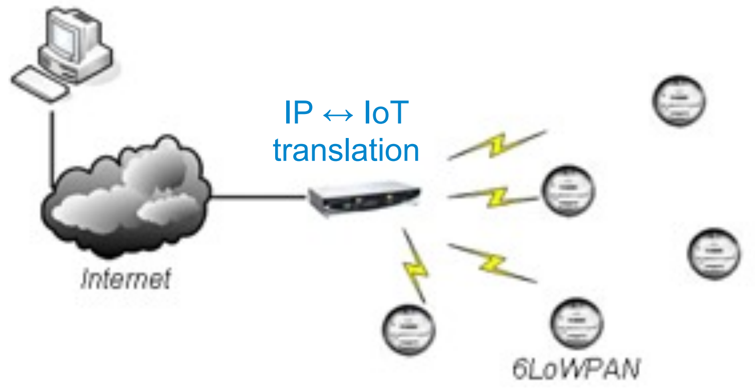


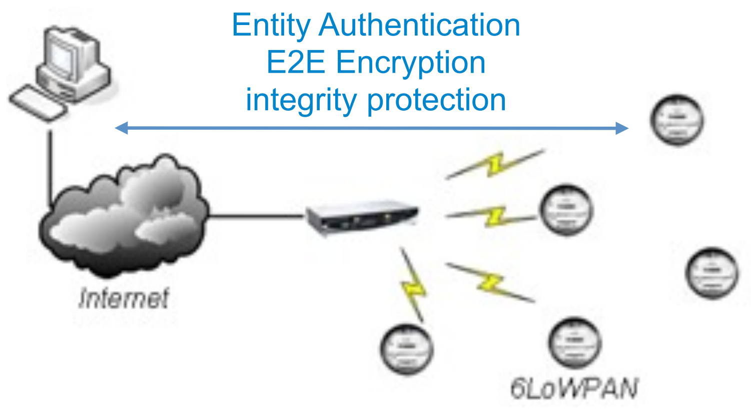


Features and requirements (1/2)

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

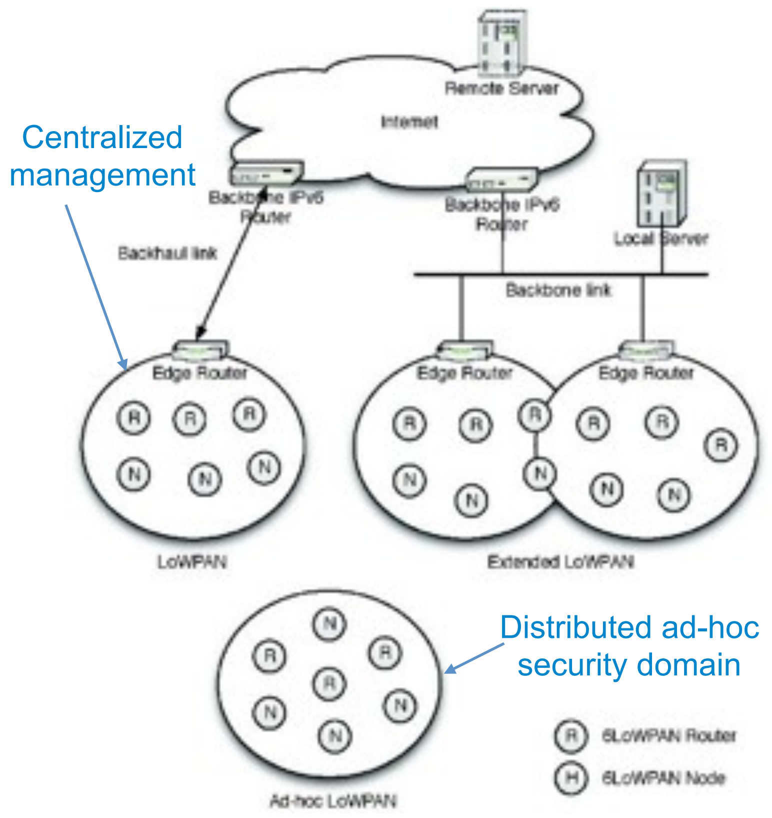




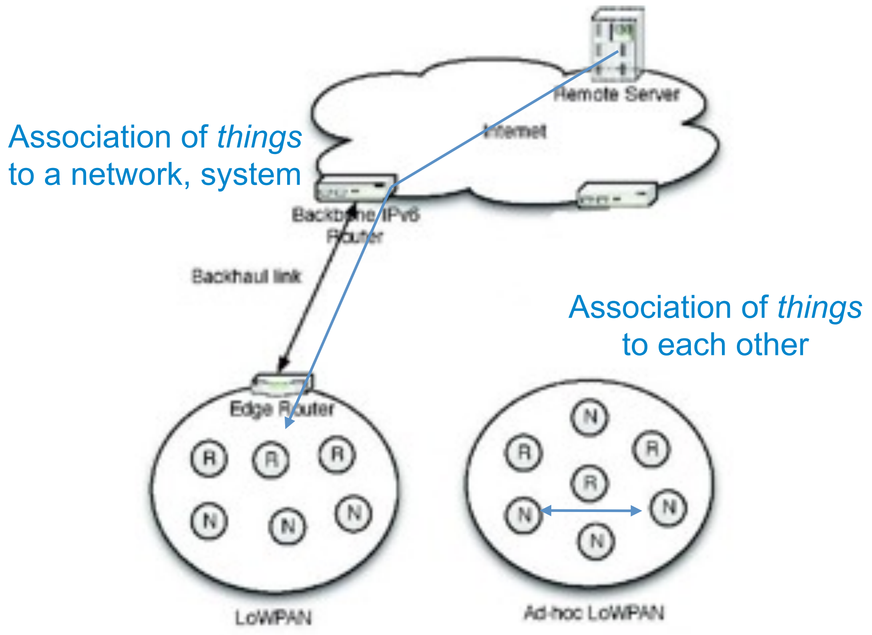


Features and requirements (2/2)

Distributed vs. Centralized architecture



Bootstrapping a thing's identity and keying materials



distributed vs. centralized

Still open

- Assessment of security mechanisms (security & operation & performance)
- Flexible security architecture and operational policies
- Selection and coordination of a default security suite
- Definition of a *standard* lightweight bootstrapping protocol

80th IETF: core WG Agenda

13:00	Introduction, Agenda, Status	Chairs (10)
13:10	1 – link-format	ZS (5)
13:15	1 – core CoAP	ZS (25)
13:40	1 – block	CB (15)
13:55	1 – observe	KH (25)
14:20	2 – basic security	ZS (20)
14:40	discovery	CB+AB (20)
15:00 retire to Wednesday, 15:10	Intro	Chairs (05)
15:15	Group Communication	AR (10)
15:25	CoAP Usage	PV (10)
15:35	HTTP Mapping	KH+AC (15)
15:50	new: SOAP; WADL; overhead; security	GM ZS AC OG (5 each)
16:10 retire		