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/

Draft name	Rev.	Dated	Status	Comments, Issues
Active:				
draft-ietf-core-coap	-01	2010-07-08	Active	2/16

Related Active Documents (not working group documents):

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

draft-bormann-coap-misc	-05	2010-07-06
draft-braun-core-compressed-ipfix	-01	2010-03-07
draft-eggert-core-congestion-control	-00	2010-06-23
draft-hartke-coap-observe	-01	2010-07-08
draft-martocci-6lowapp-building-applications	-01	2010-07-08
draft-moritz-6lowapp-dpws-enhancements	-01	2010-06-16
draft-oflynn-6lowapp-bootstrapping	-00	2010-01-27
draft-oflynn-core-bootstrapping	-01	2010-07-12
draft-rahman-core-sleeping	-00	2010-06-29
draft-shelby-core-coap	-01	2010-05-10
draft-shelby-core-coap-req	-01	2010-04-20
draft-tolle-core-ebhttp	-00	2010-03-23
draft-vanderstok-core-bc	-01	2010-07-11

CoAP Plugfest Sunday, Jul 25, 2010

- ~ 10 implementations of core-coap-01
  - most physically present, some via Internet
  - IPv6 and IPv4
- Basic interoperability done
  - message format, options encoding, transaction model
  - GET, basic link-header format
- Need to work more on specific features
  - Asynchronous transactions, subscribe (3 interoperable)
  - coap-misc features such as token, block, ... (3–4 interop.)
- Followup plugfest 1300–1530 today
  - let's just hijack the terminal room

78th IETF: core WG Agenda

09:00	Introduction, Agenda, Status	Chairs (10)
09:10	1 – core CoAP	ZS (40)
09:50	1 – Subscription option (-observe)	KH (20)
10:10	1 – coap-misc	CB (20)
10:30	1 – Congestion Control	LE (20)
10:50	1 – Sleeping Nodes	AR (15)
11:05	2 – Bootstrap approach	BS (15)
11:20	1/2 – CoAP Usage	PV (15)

78th IETF: core WG Agenda

09:00	Introduction, Agenda, Status	Chairs (10)
09:10	1 – core CoAP	ZS (40)
09:50	1 – Subscription option (-observe)	KH (20)
10:10	1 – coap-misc	CB (20)
10:30	1 – Congestion Control	LE (20)
10:50	1 – Sleeping Nodes	AR (15)
11:05	2 – Bootstrap approach	BS (15)
11:20	1/2 – CoAP Usage	PV (15)

Constrained Application Protocol (CoAP)

draft-ietf-core-coap-01

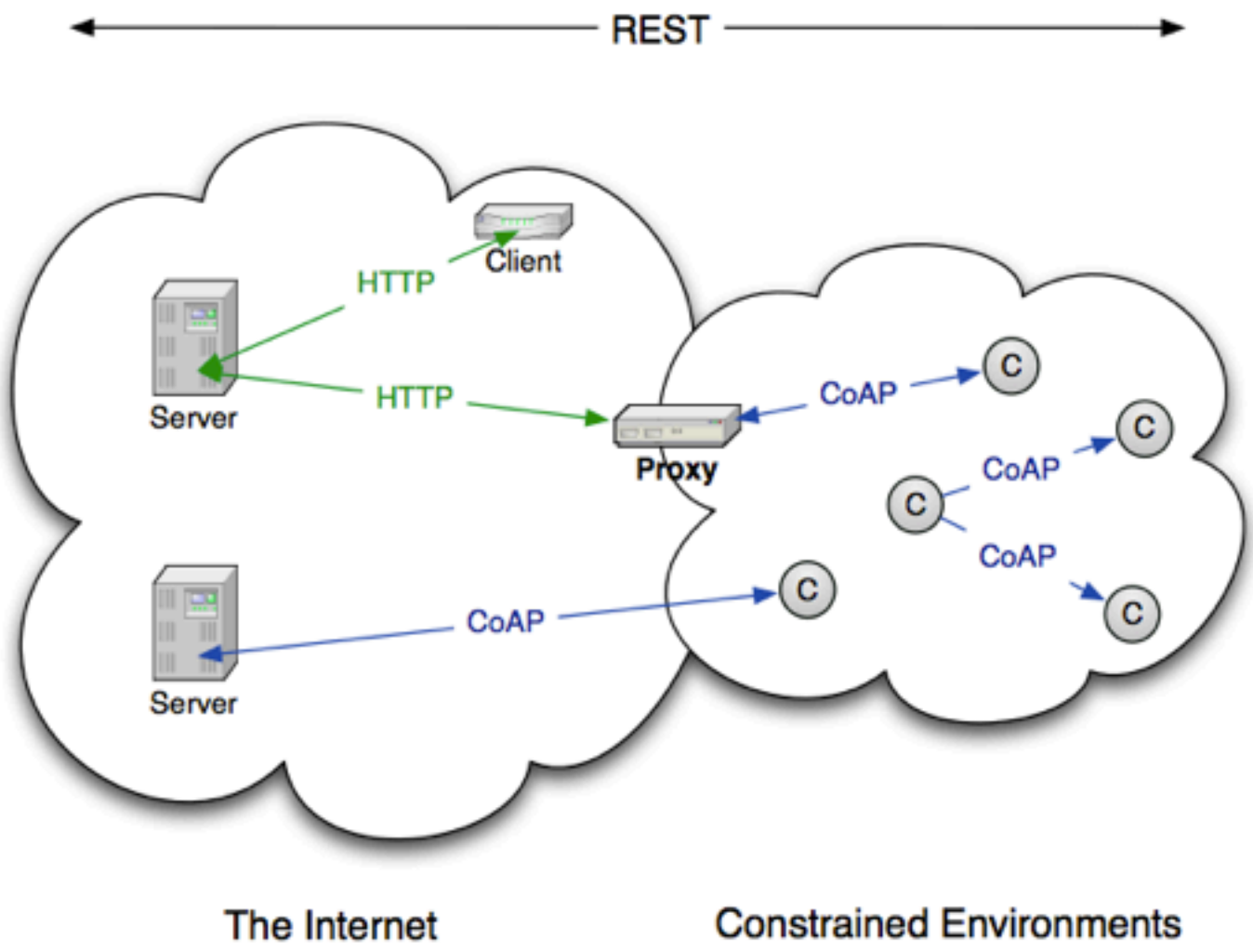
Z. Shelby, B. Frank, D. Sturek

CoRE WG, IETF-78 Maastricht

Progress since Anaheim

- coap-00 (working group document)
  - Removed TCP binding
  - Removed the magic byte header
  - Removed the Uri-Code option
  - Minor fixes and editing
- coap-01 (first complete and stable version)
  - New clean transaction model
  - Subscription moved to coap-observe-xx
  - Improved header option scheme
  - Completed all sections (proxying, HTTP mapping, discovery)
  - Minor improvements and fixes (15 tickets in total)

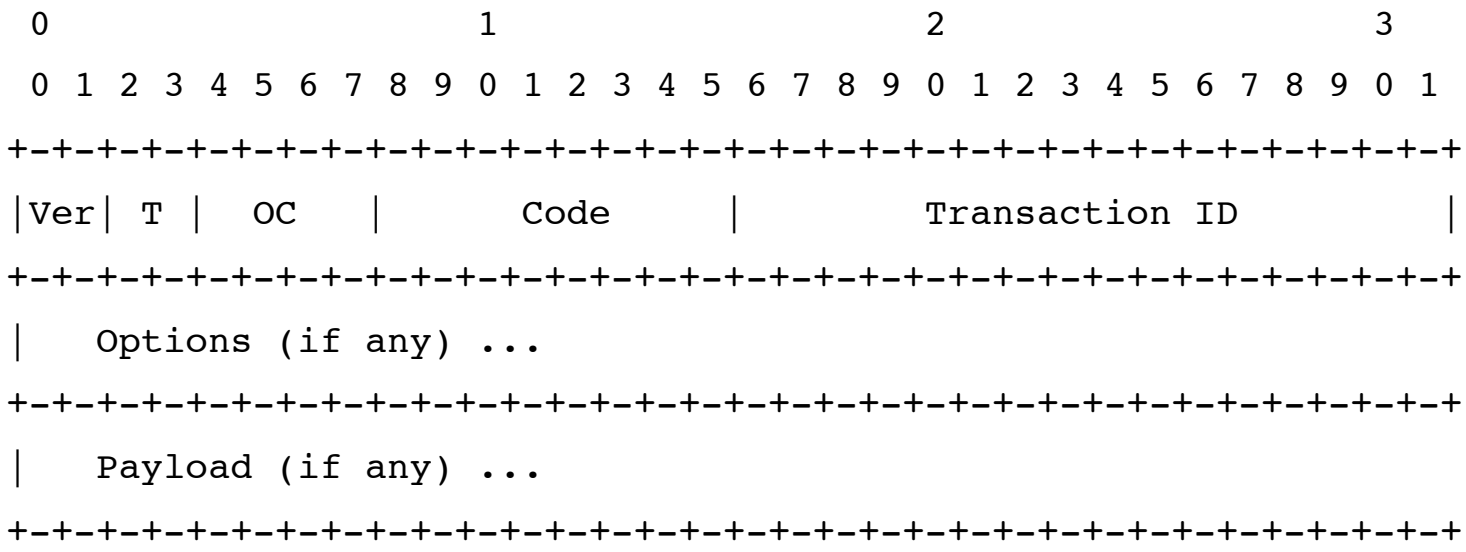
The CoRE Architecture



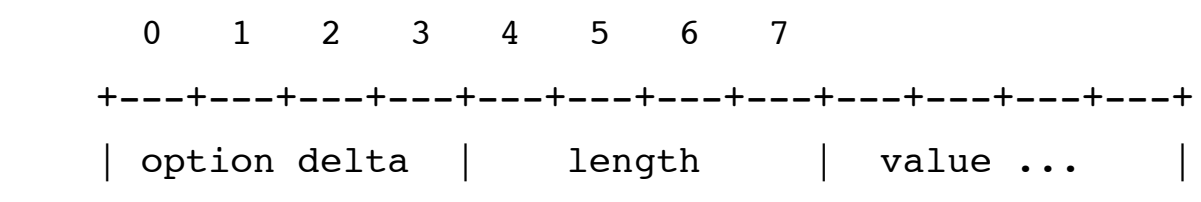
What CoAP is (and is not)

- CoAP is
  - A RESTful protocol
  - Both synchronous and asynchronous
  - For constrained devices and networks
  - Specialized for M2M applications
  - Easy to proxy to/from HTTP
- CoAP is not
  - A replacement for HTTP
  - General HTTP compression
  - Separate from the web

The Header

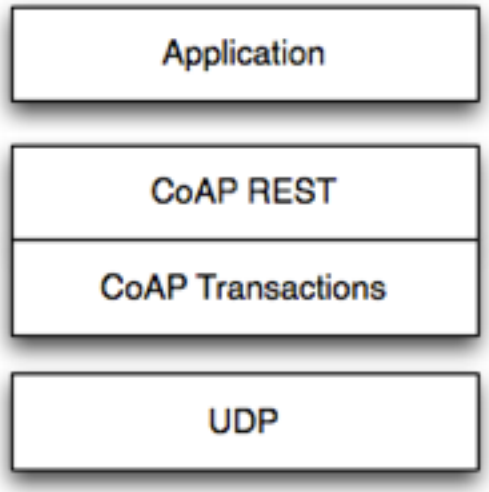


Typical Option:

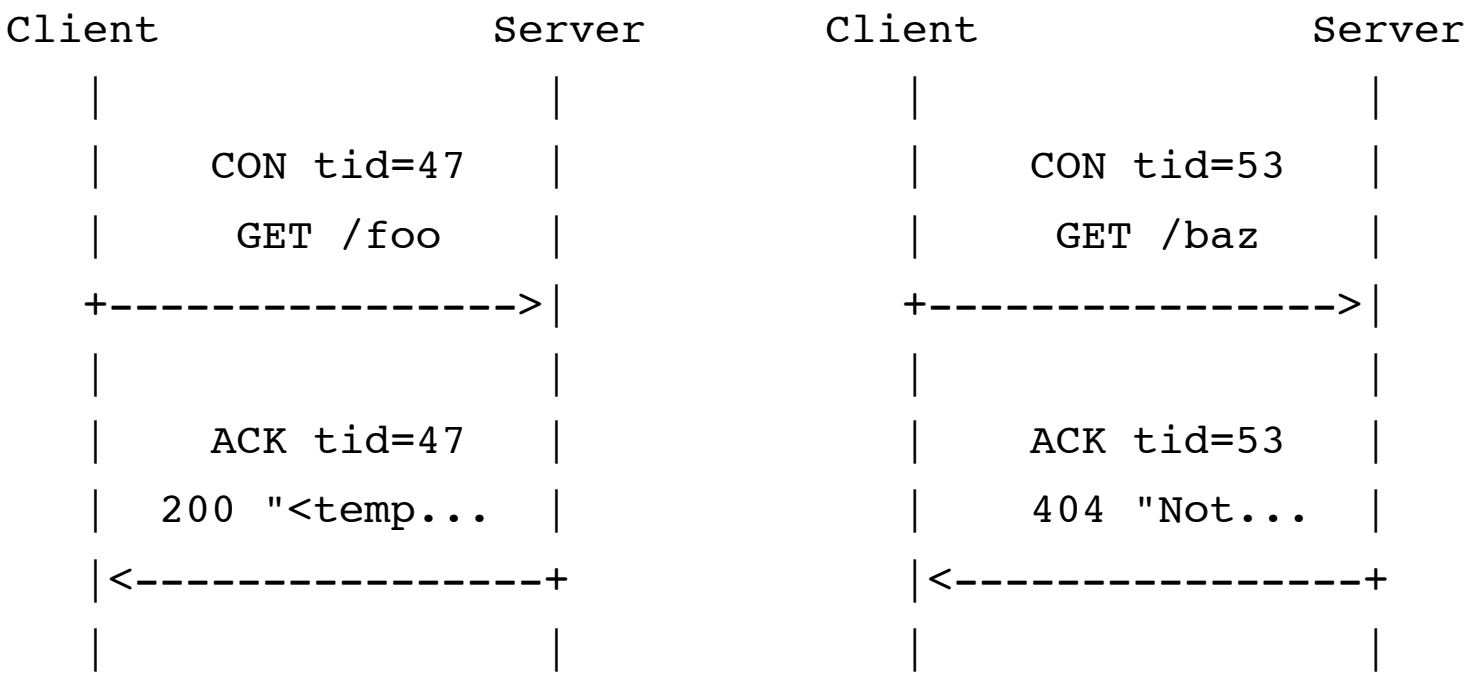


The Transaction Model

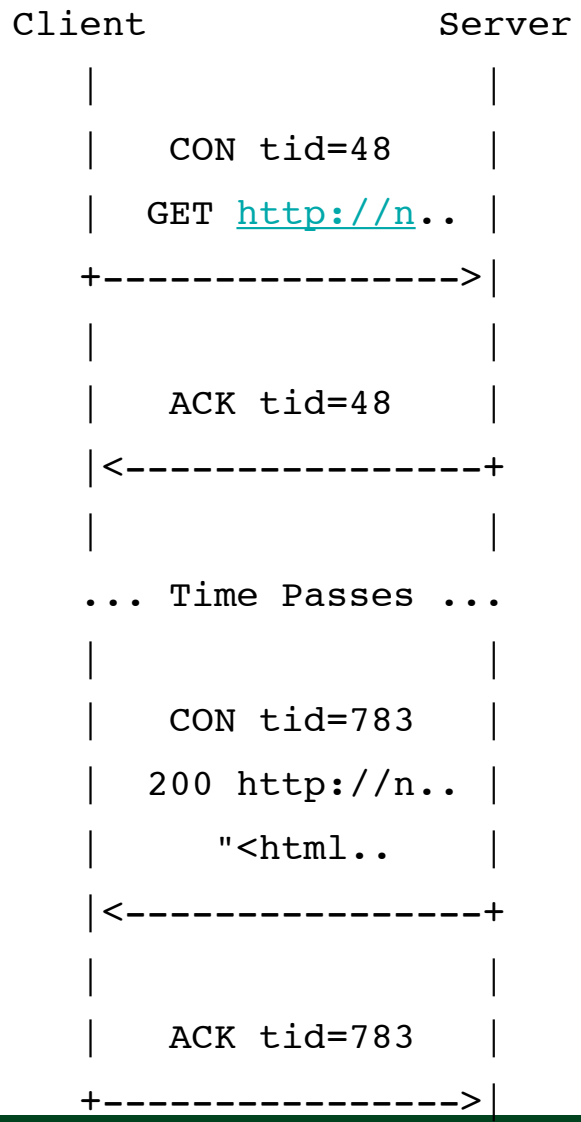
- Transport
  - CoAP is defined for UDP
- Transaction
  - Single message exchange between end-points
  - CON, NON, ACK, RST
- REST
  - Piggybacked on transaction messages
  - Method, Response Code and Options (URI, content-type etc.)



Synchronous Transaction



Asynchronous Transaction



Caching

- CoAP includes a simple caching model
  - Current only for the GET method
- Cache life
  - Controlled by the Max-Age Option
- Cache refresh and versioning
  - Using the Etag Option
- A proxy may participate in caching
  - Usually on behalf of a sleeping node

Resource Discovery

- Service Discovery
  - Leave this to e.g. DNS-SD
  - Registering _coap with dns-sd.org
- Resource Discovery
  - Retrieving the links offered by CoAP servers
  - GET /.well-known/r
  - Returns a link-header style format
    - URL, name, description, content-type, short-url, id
  - Query: GET /.well-known/r?n=Temperature

HTTP Mapping

- coap-01 defines a simple HTTP mapping for:
  - Realizing the same API over HTTP or CoAP
  - Proxying between CoAP and HTTP
- CoAP > HTTP mapping is simple
- HTTP > CoAP mapping requires checks
  - Return an error if mapping not possible
- Caching may be performed by a CoAP-HTTP proxy

I-D Proposals

- coap-observe-01
  - Subscription option integration
  - Simple HTTP poll mapping
- coap-misc-05
  - Block option integration
- coap-congestion-00

Known Bugs

- Feedback from CoRE Plugfest Sunday
  - Small link-format clarifications
  - Clarify use of Uri-Authority wrt. proxying
  - Error behavior on unknown critical option
- Reserve user-defined option (or space) (Peter)
- Fix link-format references and /.well-known (Eran)
  - Informative reference to link-format and e.g. /w/r

Open Issues

- Limit options to appear once? (Peter)
- Proposal to use 2 byte option header (Peter)
- Separate Query-Parameters option? (Peter)
- Could we remove the Uri-Scheme option? (Peter)
- How to rationalize multicast in URIs? (Peter)
- Improving discovery section (Kerry, Zach)
  - Align DNS-SD description. Only define _coap type. Instance names and TXT up to application.
  - SHOULD for CoAP server on default port
  - Multicast discovery through a no-op (CON+0)
  - Recommend unicast GETs of /.well-known/r
  - Should there be a /.well-known/host-meta?
  - Is Section 6.4 (HTTP discovery) needed?

Next Steps

- Integrate Subscription Option (via interop)?
- Integrate Block Option (via interop)?
- Close known bugs (via tickets)
- Make tickets for open issues
- Release coap-02 ~2 weeks after Maastricht

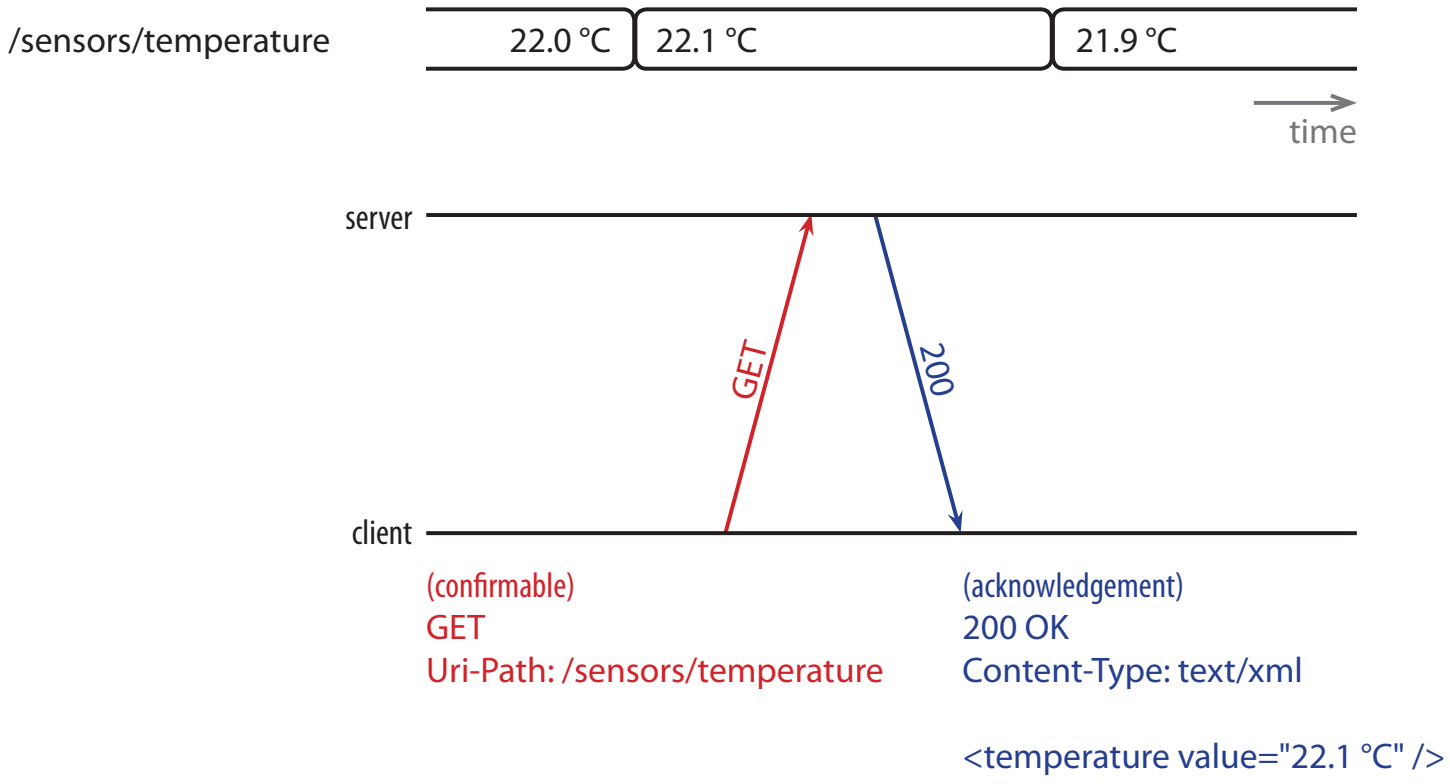
78th IETF: core WG Agenda

09:00	Introduction, Agenda, Status	Chairs (10)
09:10	1 – core CoAP	ZS (40)
09:50	1 – Subscription option (-observe)	KH (20)
10:10	1 – coap-misc	CB (20)
10:30	1 – Congestion Control	LE (20)
10:50	1 – Sleeping Nodes	AR (15)
11:05	2 – Bootstrap approach	BS (15)
11:20	1/2 – CoAP Usage	PV (15)

Observing Resources in CoAP

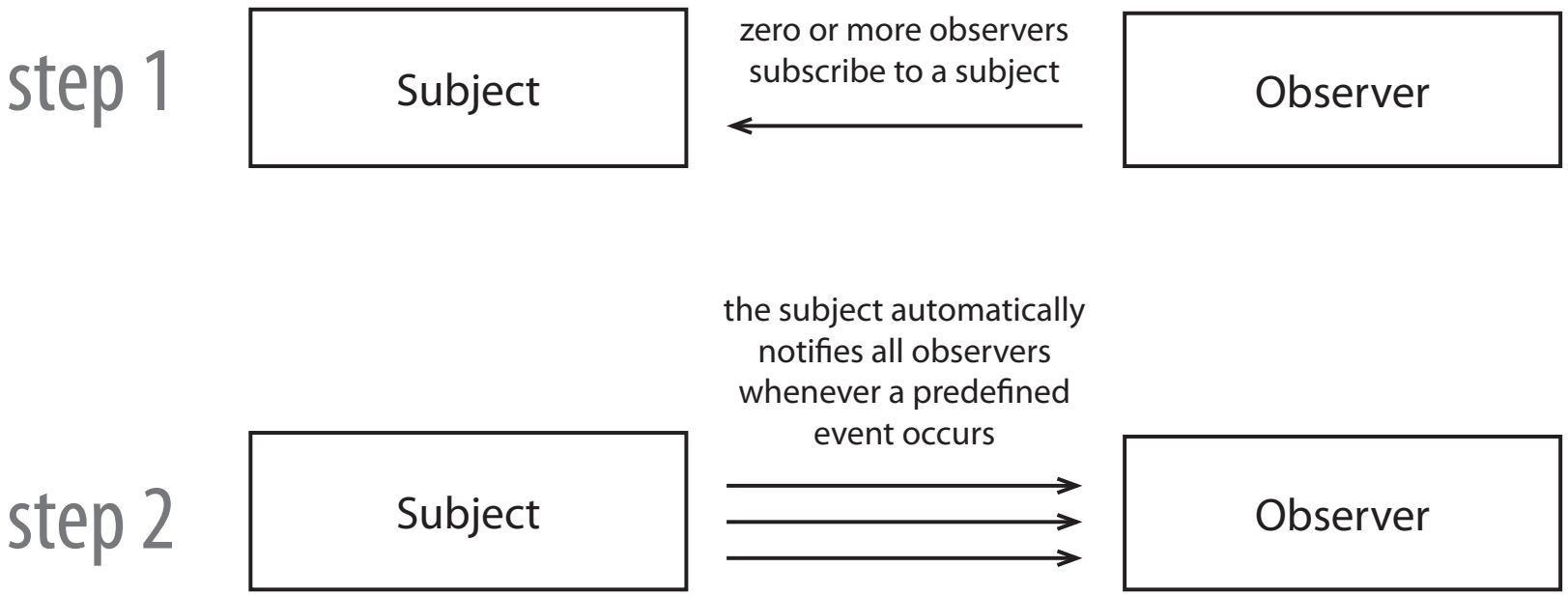
Klaus Hartke

Resources



The state of a resource can change over time. We want to observe this!

Subject/Observer Design Pattern



We can model resources as subjects!

Observers are notified whenever the state of the resource changes.

Implementing the Design Pattern in CoAP

Model resources as subjects

Observers are notified whenever the state of the resource changes

RESTful:

Observable resources are identified by URIs

Observers are notified by exchange of resource state representations

Messages are self-describing

Hypermedia as the engine of application state:
A server premediates application state transitions by providing links in resources

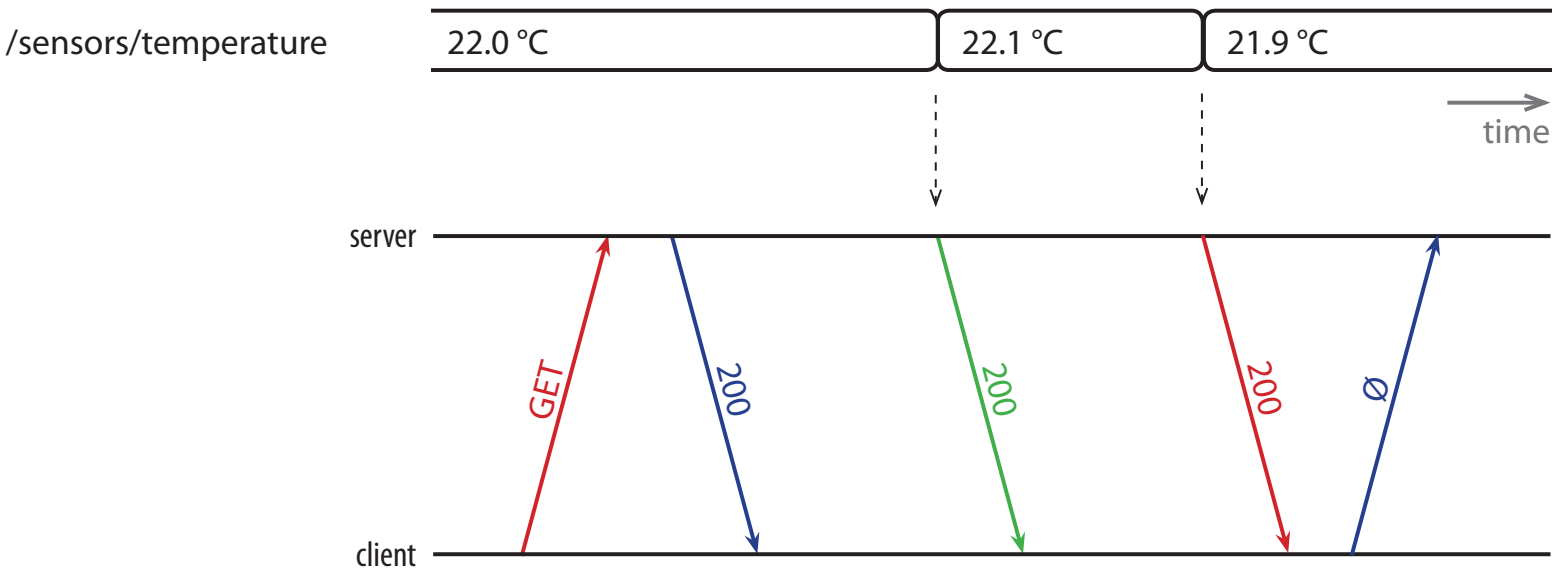
UDP-based:

Subscription and notifications are implemented by the exchange of messages

These messages arrive out of order, appear duplicated, or go missing without notice

coap-01 introduces transaction layer (CON/NON/ACK/RST)

Implementing the Design Pattern in CoAP



(confirmable)
GET
Uri-Path: /sensors/temperature
→ Subscription-Lifetime: 60s

(acknowledgement)
200 OK
Content-Type: text/xml
Subscription-Lifetime: 60s

<temperature value="22.0 °C" />

(non-confirmable)
200 OK
Uri-Path: /sensors/temperature
Content-Type: text/xml
Subscription-Lifetime: 30s

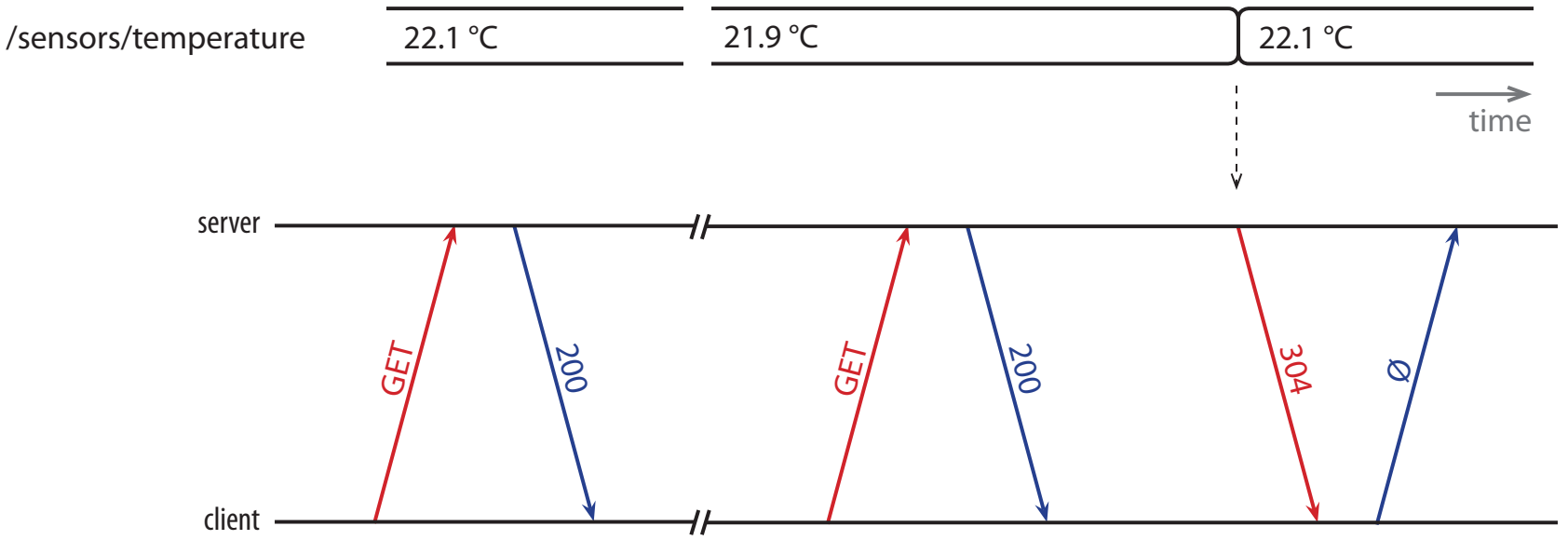
<temperature value="22.1 °C" />

(confirmable)
200 OK
Uri-Path: /sensors/temperature
Content-Type: text/xml
Subscription-Lifetime: 10s

<temperature value="21.9°C" />

(acknowledgement)
Ø

Caching



(confirmable)
GET
Uri-Path: /sensors/temperature
Subscription-Lifetime: 60s

(acknowledgement)
200 OK
Content-Type: text/xml
Subscription-Lifetime: 60s
Etag: 0xdb21ada4

<temperature value="22.1 °C" />

(confirmable)
GET
Uri-Path: /sensors/temperature
Subscription-Lifetime: 60s
→ Etag: 0xdb21ada4

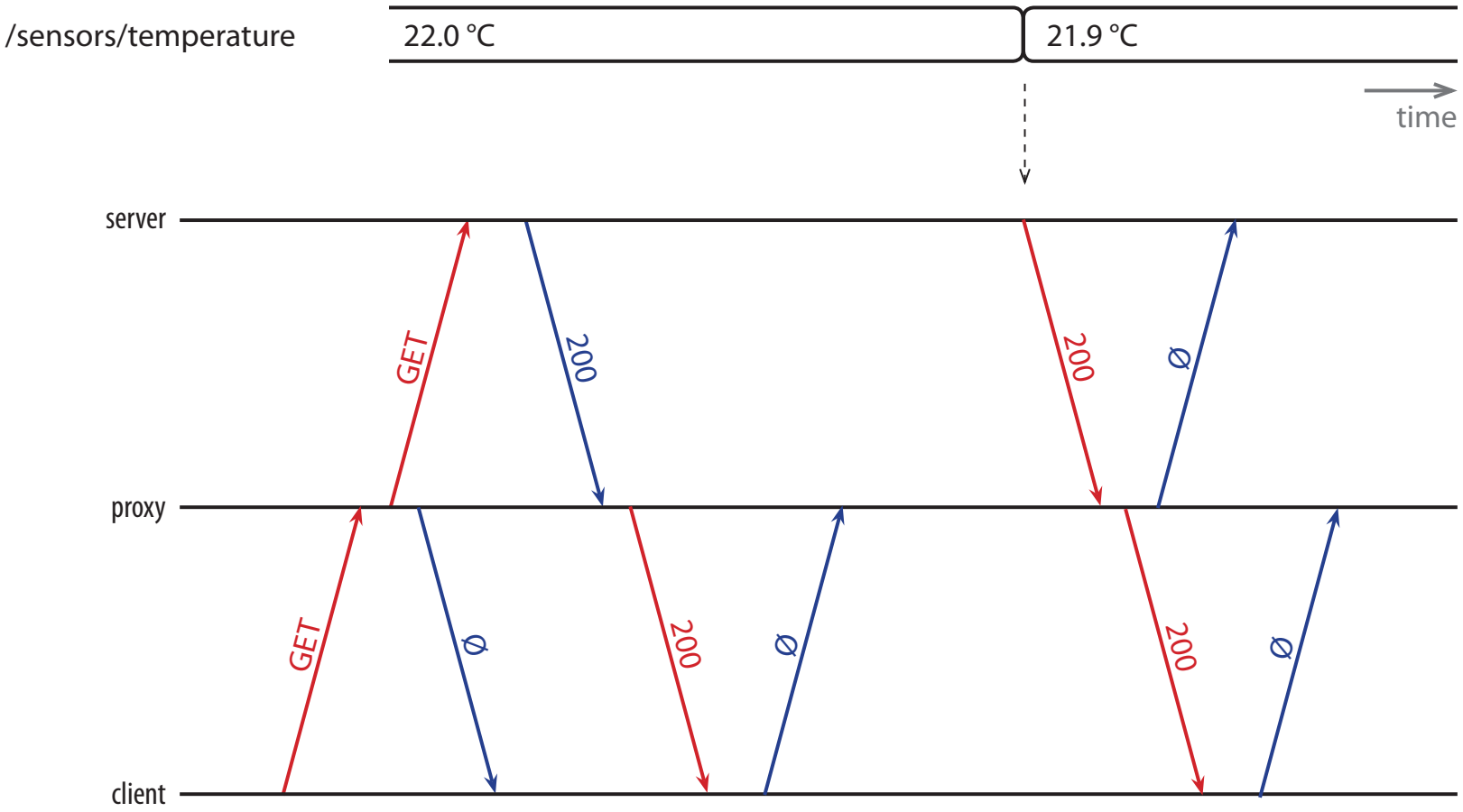
(acknowledgement)
200 OK
Content-Type: text/xml
Subscription-Lifetime: 60s
Etag: 0x22bd01c4

<temperature value="21.9 °C" />

(confirmable)
304 Not Modified
Uri-Path: /sensors/temperature
Etag: 0xdb21ada4
Subscription-Lifetime: 10s

(acknowledgement)
Ø

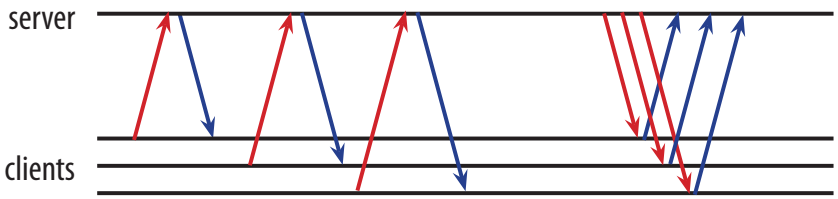
Proxying


/sensors/temperature
22.0 °C
21.9 °C
time
server
proxy
client
GET
200
Ø
GET
200
Ø
200
Ø
200
Ø

Multiple observers

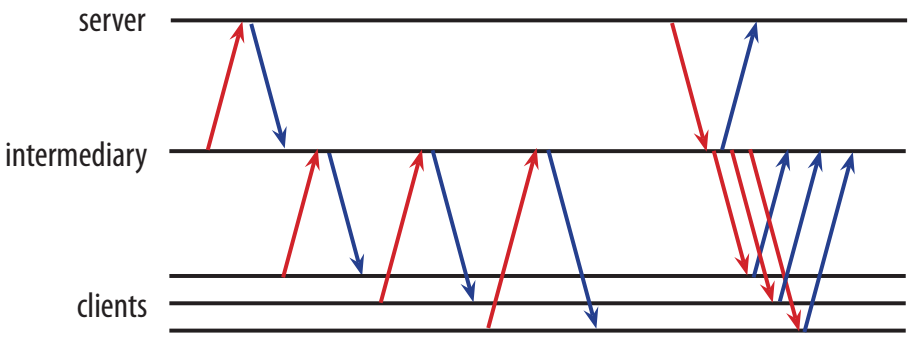
option 1

simply subscribe multiple observers to a resource



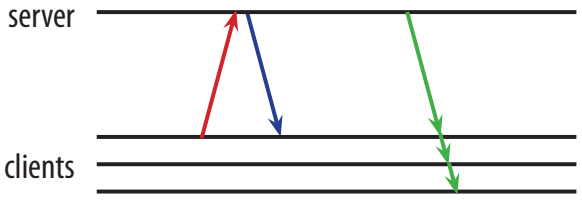
option 2

subscribe multiple observers to an intermediary node that maintains a single subscription to a resource



option 3

subscribe an IPv6 multicast group to a resource



(confirmable)
GET
Uri-Path: /sensors/temperature
Subscription-Lifetime: indefinite
Reply-To: [ffxx::xxxx]:61616

Summary

RESTful sub/not mechanism based on well-known design pattern

Observing resources is fun!

Once you start looking for observable things, you see them everywhere!

All prerequisites already in coap-01

Concrete proposal that works well with caching, proxying and many observers

Running code

2 servers & 3 client implementations

Next steps

Nail down exact semantics of Subscription-Lifetime option

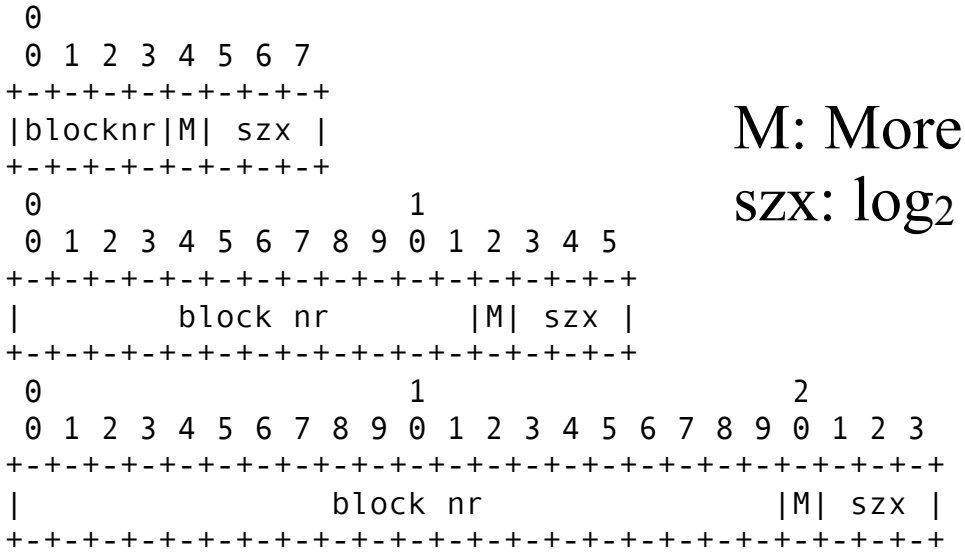
Check interactions with other CoAP features

78th IETF: core WG Agenda

09:00	Introduction, Agenda, Status	Chairs (10)
09:10	1 – core CoAP	ZS (40)
09:50	1 – Subscription option (-observe)	KH (20)
10:10	1 – coap-misc	CB (20)
10:30	1 – Congestion Control	LE (20)
10:50	1 – Sleeping Nodes	AR (15)
11:05	2 – Bootstrap approach	BS (15)
11:20	1/2 – CoAP Usage	PV (15)

The block option

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



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

Decisions:

- Block size is power of 2
- $16 \leq$ Block size ≤ 2048

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**

Accept Option

- What media type would I want to get?
- Cf. Accept: in HTTP
- Option value: sequence of bytes, each byte is a Content-Type

- Alternative: repeatable Content-Type

- Accept is ELECTIVE

TeRIs

- URI encoding schemes not very useful (25 % gain)
- Better: Provide shorter, temporary RIs
  - e.g., in a block transfer: provide TeRI with block 0

- TeRI: 1 byte duration (lifetime), n bytes identifier
- TeRI is ELECTIVE
  - Oops

Token

- **Provide a way to relate a response to a request**
  - **beyond single-transaction TID**
- **Token is ELECTIVE**

Uri-Authority-Binary

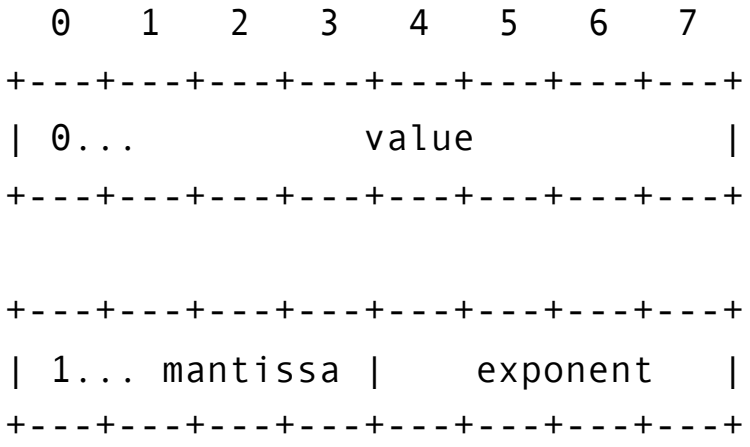
- IPv4, IPv6 IID, or IPv6 address (default: dest. address)
- optional port number (default: dest. port)
- detect which it is by length
  - 2, 4, 6, 8, 10, 16, 18

Payload Length

- **CoAP assumes known datagram length**
  - **no need to explicitly give payload length**
- **How to aggregate multiple messages in one packet?**
  - **do explicitly give payload length**
- **Payload-Length is CRITICAL**

Duration Data Type (1)

- Many Options need a Duration (length of timespan)
- Resolution mostly 1 second
- can use variable-length integer
- often, there is no need for this complexity



Duration Data Type (2)

- **Extremely easy to decode**
- `#define DECODE_8_4(r) (r < HIBIT ? r : (r & MMASK) << (r & EMASK))`
- **Reasonably easy to encode**
  - **two directions of rounding**
- **Range: 0..127 s exact, 128 s .. 84d 22:53:52 s (–12.5 %)**
  - **Do we need more than 12 weeks?**
- **Reserve 0xFF for “indefinite”**

78th IETF: core WG Agenda

09:00	Introduction, Agenda, Status	Chairs (10)
09:10	1 – core CoAP	ZS (40)
09:50	1 – Subscription option (-observe)	KH (20)
10:10	1 – coap-misc	CB (20)
10:30	1 – Congestion Control	LE (20)
10:50	1 – Sleeping Nodes	AR (15)
11:05	2 – Bootstrap approach	BS (15)
11:20	1/2 – CoAP Usage	PV (15)

78th IETF: core WG Agenda

09:00	Introduction, Agenda, Status	Chairs (10)
09:10	1 – core CoAP	ZS (40)
09:50	1 – Subscription option (-observe)	KH (20)
10:10	1 – coap-misc	CB (20)
10:30	1 – Congestion Control	LE (20)
10:50	1 – Sleeping Nodes	AR (15)
11:05	2 – Bootstrap approach	BS (15)
11:20	1/2 – CoAP Usage	PV (15)

Sleeping and Multicast Considerations for CoAP

Akbar Rahman
Juan Carlos Zúñiga
Guang Lu

IETF 78, July 2010
http://tools.ietf.org/html/draft-rahman-core-sleeping-00

Introduction

- We further analyze the following CoAP requirements related to “sleeping nodes” and “multicast”:
  - REQ 3: The ability to deal with sleeping nodes. Devices may be powered off at any point in time but periodically "wake up" for brief periods of time.
  - REQ 4: Protocol must support the caching of recent resource requests, along with caching subscriptions to sleeping nodes.
  - REQ 9: CoAP will support a non-reliable IP multicast message to be sent to a group of Devices to manipulate a resource on all the Devices simultaneously. The use of multicast to query and advertise descriptions must be supported, along with the support of unicast responses.

Figure 1- Use Case of Originating CoAP Transaction and Sleeping Node

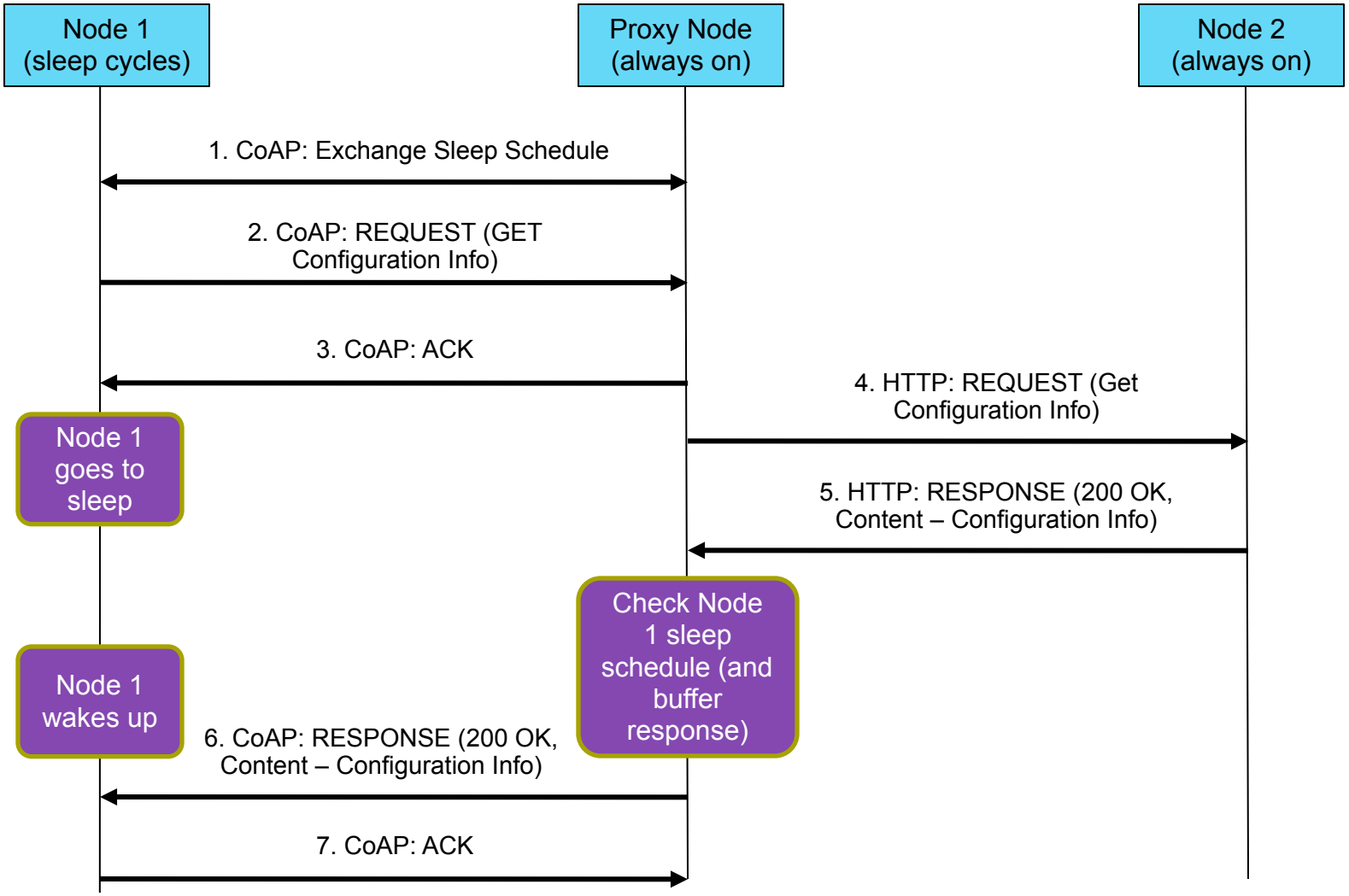
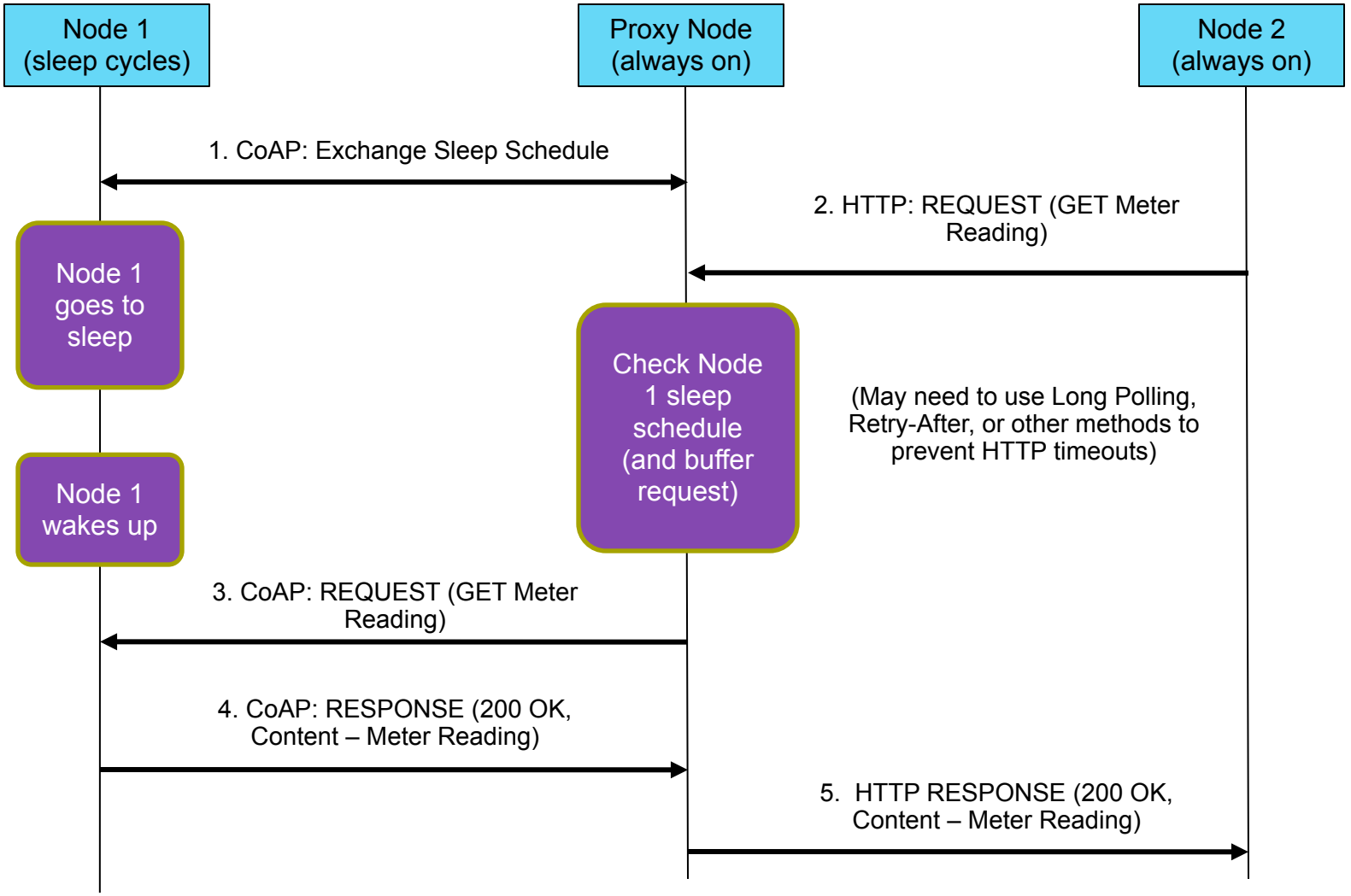


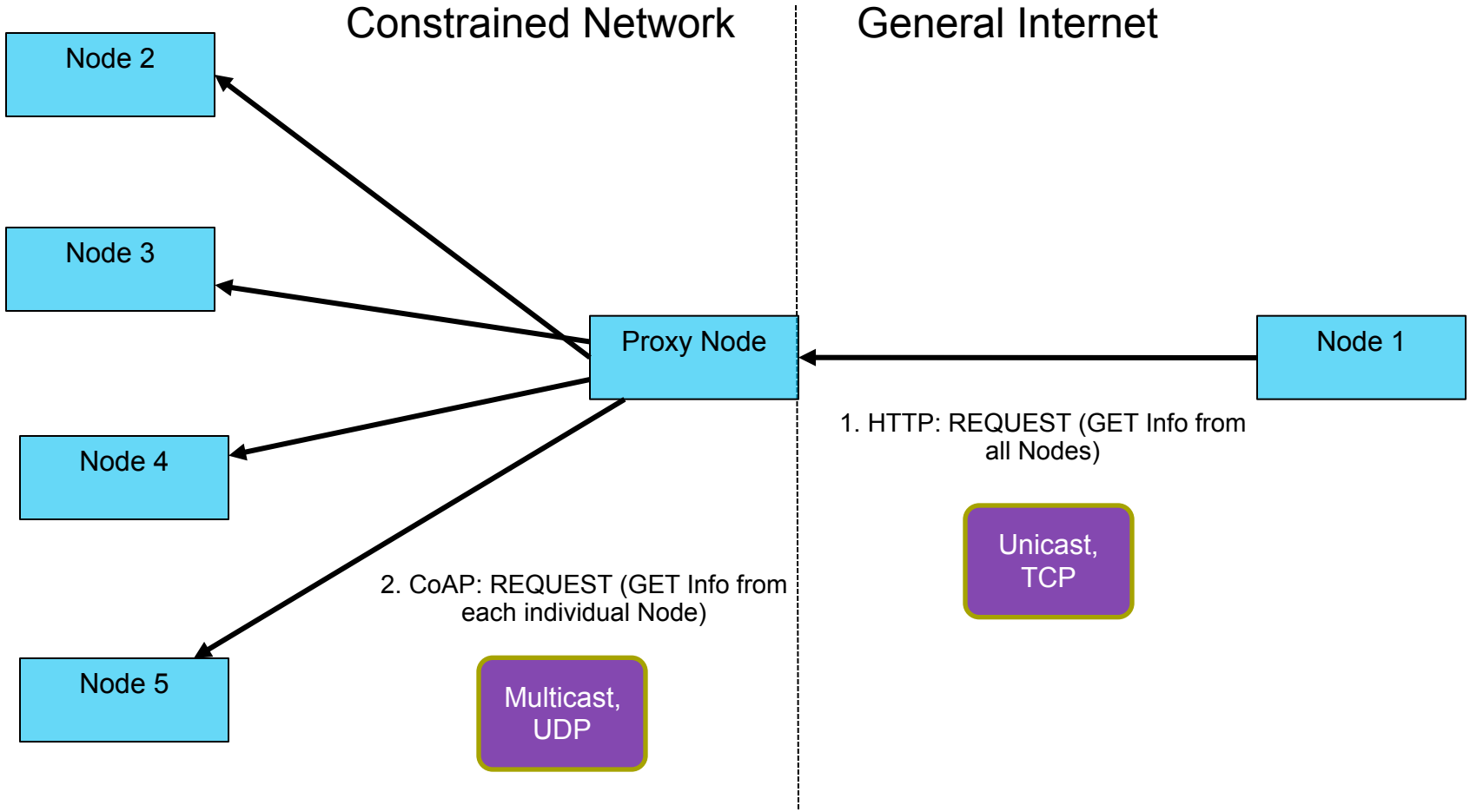
Figure 2- Use Case of Terminating CoAP Transaction and Sleeping Node



Further considerations for Sleeping Nodes

- What format should the sleeping schedule be in? And how do the nodes synchronize?
  - Wireless technologies typically support procedures for the above:
    - For example, the proposed 802.15.4e draft supports detailed PHY/MAC layer procedures for sleeping schedule and synchronization
    - So, one approach for CoAP could be to leverage and extend upon the PHY/MAC layer synchronization and scheduling (e.g. for the CoAP layer in the Proxy to have an API to these lower layers to retrieve the required information)

Figure 3- Multicast Problem Scenario



Further considerations for Multicast

- What would the URI look like that the client uses on the proxy?
- How would the proxy relay back the multitude of responses?
- How would overall congestion control work?
- What happens if some of the CoAP nodes are sleeping?

Conclusions (1/2)

- For CoAP to handle sleeping nodes:
  - If the proxy node has an updated schedule for each sleep node
  - Then the proxy node can more optimally buffer responses destined for sleeping nodes as well as service incoming requests on behalf of sleeping nodes via intercept caching

Conclusions (2/2)

- For CoAP to handle multicast:
  - HTTP runs on TCP in the general Internet
  - And IP multicast does not support TCP
  - The proxy node in the constrained network needs to have functionality to support interworking between multicast (in the constrained network) and unicast (in the Internet)

Next Steps

- If the WG agrees, then we can update our draft to move beyond the problem statement stage and move into the detailed solutions for both sleeping node and multicast support for CoAP

78th IETF: core WG Agenda

09:00	Introduction, Agenda, Status	Chairs (10)
09:10	1 – core CoAP	ZS (40)
09:50	1 – Subscription option (-observe)	KH (20)
10:10	1 – coap-misc	CB (20)
10:30	1 – Congestion Control	LE (20)
10:50	1 – Sleeping Nodes	AR (15)
11:05	2 – Bootstrap approach	BS (15)
11:20	1/2 – CoAP Usage	PV (15)

What is CORE chartered to do?

- Security, particularly **keying of new Devices**, is very challenging [...]. The WG will work to select approaches to **security bootstrapping** which are realistic [...]. To ensure that **any two nodes can join** together, all nodes must implement at least one **universal bootstrapping method**.
- Security can be achieved using either **session security or object security**. For both object and session security, the WG will work with the security area to select appropriate security framework and protocol as well as selecting a minimal required to implement cipher suite. CoAP will **initially** look at **CMS** (RFC 5652), **TLS/DTLS**, and **EAP**.

Bootstrapping

Colin O’flynn
Behcet Sarikaya (presenter)
Robert Cragie

Overview

- Definition of Bootstrapping
- Problems Faced
- Existing Solutions
- Proposed Framework
- Fitting In with CoAP

Bootstrapping – What is it?

- The magic that takes a network from a box of nodes to a fully functioning network

Bootstrapping – What is it not?

- Does not replace service or resource discovery
  - Bootstrapping is finished when normal network operation can begin, at which point service or resource discovery can occur

Bootstrapping – Problems

- Merging Networks
  - If a node is already on a network, and the user wishes this node to join another network, what happens?
- Node Mobility
- Resource Constraints
  - Computational, Power, Size, and Price
- User Interface
  - Wide range of nodes: from full graphical LCD to no user interface
- Security

Existing Solutions

- Examples of solutions to these problems exist in several standards, such as :
  - WiFi Protected Setup (WPS)
  - Bluetooth
  - Wireless USB
- Typically defined for too narrow an application-space for CoRE though. As CoRE nodes span the range from:
  - Tiny parasitic power devices to wall-powered nodes
  - 8-bit microcontrollers to 32-bit processors
  - Low to High security requirements (ie: light switch vs. smart meter)

Proposed Architecture

- **Communications Channel**: Used during normal network operation (e.g.: 802.15.4)
  - **Control Channel**: Used for bootstrapping only
- **Supported Channels**: IEEE 802.15.4, Power-line Communications, IRDA, RFID, Simple physical link, cellular, Ethernet, IPv6
- **User Interface**: Defines what the user controls the node with (e.g.: pushbutton, keyboard)
- **Bootstrap Profile**: Defines information exchanged during bootstrapping (e.g.: channel settings, encryption keys)

Proposed Architecture

- **Security Method**: Defines supported security methods for bootstrapping
- **Available Methods**:
  - None
  - EAP Methods, e.g. EAP TLS v1.2, etc.
  - Asymmetric with User Authentication, Followed by Symmetric
  - Asymmetric with Certificate Authority, Followed by Symmetric
  - Cryptographically Generated Address Based Address Ownership Verification

Proposed Architecture

- **Bootstrap Protocol**: Actual messages exchanged for bootstrapping
- The protocol is likely a wrapper on existing authentication functions, e.g. EAP
- Bootstrap protocol will negotiate allowable standards between nodes
  - When a TV is joining a remote control, the protocol must understand that the remote control has very limited resources even though TV may have a complex interface available

Fitting in with CoAP

- Bootstrapping requires input from other layers to work!
  - User needs to select networks/nodes to join
  - Node may automatically join networks based on available services
    - Bootstrapping should NOT duplicate service discovery, but work with the proper layers / standards
- Bootstrapping difficult to implement “cleanly”

Next Steps

- Feedback from requirements of different users, e.g. Zigbee IPSTACK group
- Decide on standards which bootstrapping will use
- Fit bootstrapping and CoAP together
- Finish the documentation as an architecture document
- Bootstrapping solution document in the next stage

78th IETF: core WG Agenda

09:00	Introduction, Agenda, Status	Chairs (10)
09:10	1 – core CoAP	ZS (40)
09:50	1 – Subscription option (-observe)	KH (20)
10:10	1 – coap-misc	CB (20)
10:30	1 – Congestion Control	LE (20)
10:50	1 – Sleeping Nodes	AR (15)
11:05	2 – Bootstrap approach	BS (15)
11:20	1/2 – CoAP Usage	PV (15)

CoAP Utilization for Building Control

draft-vanderstok-core-bc-01

Peter van der Stok
Kerry Lynn

July 28, 2010

Motivating questions

Grouping of nodes
Service/Resource discovery
Handling of legacy
Size of uri
Battery-less devices
Multicast specification

Grouping of nodes

Logical groups coincide with hierarchical building structure

- Lights in a room (activated by PIR)
- Convectors at a floor (controlled by floor temperature)
- On/Off switching in building (controlled by clock)

Example authorities

- //all.bldg6
- //all.west.bldg6
- //all.floor1.west.bldg6
- //all.bu036.floor1.west.bldg6

Service/resource discovery

Nodes are grouped. Not resources
- all resources on a node belong to groups to which node belongs

One coap service assumed per node
Use DNS-SD to discover the coap service
Use DNS-SD to discover to coap service groups

A node returns its resources according to coap resource discovery

Equivalent with BACnet Who-is and Who has.

Groups/names are building/owner specific
Resource naming requires standardization for interoperability

Handling of legacy

Silos use their individual networking standards:
DALI, BACnet, LONtalk, KNX, Zigbee Device Objects (ZDO), etc.

Assumed phased introduction of CoAP to building control:
1. Phase 1: CoAP transports legacy standard
2. Phase 2: CoAP transports building control naming standard

Phase 1 example

DALI command:
Switch on

CoAP message Confirmable
PUT method
Mime type: /application/DALI
DALI Switch on

Unpack message
DALI invoked
Light switched on

Size of uri

Authority of URI is resolved to single a unicast or multicast address, plus port.

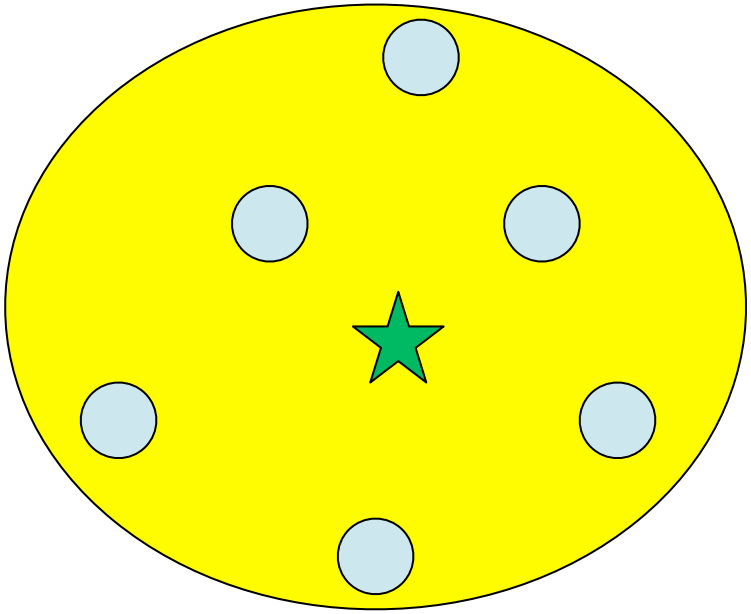
Path specifies resource: standard dependent (e.g. single 16 bit value)

Battery-less device

Battery less node sends at (ir)regular intervals, and sleeps

actuator node is always on and receives

controller node, is always on, receives and redistributes

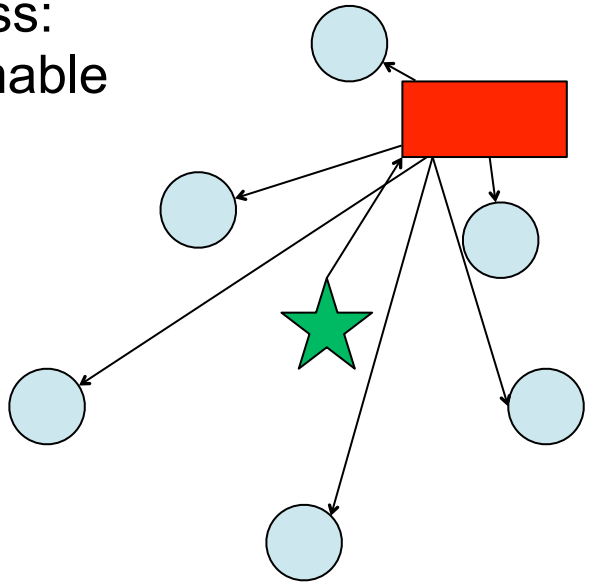


Multicast scope

From battery-less:

Non confirmable

Put



Multicast specification

Scope defined by group (hierarchical building structure)

Specification:

- Validity
- Integrity
- Agreement
- Timeliness
- Ordering?